Conception et analyse des biopuces à ADN en environnements parallèles et distribués

Faouzi Jaziri

Université Blaise Pascal – Clermont II

Ecole Doctorale des Sciences pour l'Ingénieur

Thèse

pour obtenir le grade de

Docteur d'Université

(Spécialité: Informatique)

Présentée par

Faouzi JAZIRI

Conception et analyse des biopuces à ADN en environnements parallèles et distribués

Date de soutenance prévue : 23 Juin 2014

Membres du jury

Rapporteurs:	**Abdoulaye Baniré Diallo, Professeur, Université du Québec**
	Robert Duran, Professeur, Université de Pau et des Pays de l'Adour
Examinateurs:	**Vincent Breton, Directeur de recherche, CNRS**
	Eric Innocenti, Maître de conférences, Université de Corse Pasquale Paoli
Directeurs de thèse:	**David Hill, Professeur, Université Blaise Pascal**
	Pierre Peyret, Professeur, Université d'Auvergne
Invité:	**Eric Peyretaillade, Maître de conférences, Université d'Auvergne**

Laboratoires d'accueil:

Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)
EA Conception Ingénierie et Développement de l'Aliment et du Médicament (CIDAM)
Laboratoire Microorganismes: Génome et Environnement (LMGE)

Résumé

Les microorganismes constituent la plus grande diversité du monde vivant. Ils jouent un rôle clef dans tous les processus biologiques grâce à leurs capacités d'adaptation et à la diversité de leurs capacités métaboliques. Le développement de nouvelles approches de génomique permet de mieux explorer les populations microbiennes. Dans ce contexte, les biopuces à ADN représentent un outil à haut débit de choix pour l'étude de plusieurs milliers d'espèces en une seule expérience. Cependant, la conception et l'analyse des biopuces à ADN, avec leurs formats de haute densité actuels ainsi que l'immense quantité de données à traiter, représentent des étapes complexes mais cruciales. Pour améliorer la qualité et la performance de ces deux étapes, nous avons proposé de nouvelles approches bioinformatiques pour la conception et l'analyse des biopuces à ADN en environnements parallèles. Ces approches généralistes et polyvalentes utilisent le calcul haute performance (HPC) et les nouvelles approches du génie logiciel inspirées de la modélisation, notamment l'ingénierie dirigée par les modèles (IDM) pour contourner les limites actuelles.

Nous avons développé PhylGrid 2.0, une nouvelle approche distribuée sur grilles de calcul pour la sélection de sondes exploratoires pour biopuces phylogénétiques. Ce logiciel a alors été utilisé pour construire PhylOPDb: une base de données complète de sondes oligonucléotidiques pour l'étude des communautés procaryotiques. MetaExploArrays qui est un logiciel parallèle pour la détermination de sondes sur différentes architectures de calcul (un PC, un multiprocesseur, un cluster ou une grille de calcul), en utilisant une approche de méta-programmation et d'ingénierie dirigée par les modèles a alors été conçu pour apporter une flexibilité aux utilisateurs en fonction de leurs ressources matériel. PhylInterpret, quant à lui est un nouveau logiciel pour faciliter l'analyse des résultats d'hybridation des biopuces à ADN. PhylInterpret utilise les notions de la logique propositionnelle pour déterminer la composition en procaryotes d'échantillons métagénomiques. Enfin, une démarche d'ingénierie dirigée par les modèles pour la parallélisation de la traduction inverse d'oligopeptides pour le design des biopuces à ADN fonctionnelles a également été mise en place.

Mots-clés: Bioinformatique, Biopuces à ADN, Sélection de sondes, Analyse des biopuces, Calcul intensif, Ingénierie dirigée par les modèles (IDM).

Abstract

Microorganisms represent the largest diversity of the living beings. They play a crucial role in all biological processes related to their huge metabolic potentialities and their capacity for adaptation to different ecological niches. The development of new genomic approaches allows a better knowledge of the microbial communities involved in complex environments functioning. In this context, DNA microarrays represent high-throughput tools able to study the presence, or the expression levels of several thousands of genes, combining qualitative and quantitative aspects in only one experiment. However, the design and analysis of DNA microarrays, with their current high density formats as well as the huge amount of data to process, are complex but crucial steps. To improve the quality and performance of these two steps, we have proposed new bioinformatics approaches for the design and analysis of DNA microarrays in parallel and distributed environments. These multipurpose approaches use high performance computing (HPC) and new software engineering approaches, especially model driven engineering (MDE), to overcome the current limitations.

We have first developed PhylGrid 2.0, a new distributed approach for the selection of explorative probes for phylogenetic DNA microarrays at large scale using computing grids. This software was used to build PhylOPDb: a comprehensive 16S rRNA oligonucleotide probe database for prokaryotic identification. MetaExploArrays, which is a parallel software of oligonucleotide probe selection on different computing architectures (a PC, a multiprocessor, a cluster or a computing grid) using meta-programming and a model driven engineering approach, has been developed to improve flexibility in accordance to user's informatics resources. Then, PhylInterpret, a new software for the analysis of hybridization results of DNA microarrays. PhylInterpret uses the concepts of propositional logic to determine the prokaryotic composition of metagenomic samples. Finally, a new parallelization method based on model driven engineering (MDE) has been proposed to compute a complete backtranslation of short peptides to select probes for functional microarrays.

Keywords: Bioinformatics, DNA microarrays, Probe design, DNA MicroArray Data Analysis, High performance computing (HPC), Model driven engineering (MDE).

Remerciements

بسم الله الرحمن الرحيم

Je souhaite remercier d'abord les différents directeurs d'unités, Alain Quillot (Laboratoire d'Informatique et de Modélisation et Optimisation des Système LIMOS – UMR CNRS 6158), Christian Amblard (Laboratoire Microorganismes: Génome et Environnement LMGE – UMR CNRS 6023) et Monique Alric (EA Conception Ingénierie et Développement de l'Aliment et du Médicament CIDAM - EA 4678), pour m'avoir accueilli au sein de leurs laboratoires. Je vous remercie pour votre accueil et pour m'avoir permis de travailler dans de bonnes conditions.

Mes remerciements les plus sincères vont ensuite à Abdoulaye Baniré Diallo et Robert Duran pour m'avoir fait l'honneur d'accepter d'être rapporteurs de cette thèse ainsi qu'à Vincent Breton et Eric Innocenti pour avoir accepté d'examiner mon travail et faire partie de mon jury de thèse. Je voudrais également remercier la région d'Auvergne pour le financement des travaux de ma thèse.

Mes remerciements les plus chaleureux vont à mes deux directeurs de thèse, David Hill et Pierre Peyret, pour avoir encadré mes travaux de recherche durant cette thèse jusqu'à leur aboutissement. Je te remercie Benny pour ton aide, tes conseils et ton soutien dans les moments difficiles. Merci pour m'avoir toujours poussé à donner le meilleur de moi-même. Merci pour ton côté humain qui me touche depuis plusieurs années maintenant. Je ne te remercierai jamais assez pour tout ce que tu as fait pour moi. Et je ne peux pas oublier de remercier ta femme Anne. Merci Anne pour ton grand cœur et ta gentillesse. Mes remerciements vont aussi pour « Benny sénior » pour les relectures d'anglais! Je souhaite rendre hommage à Pierre pour son soutien, sa confiance et son aide précieuse. Merci pour m'avoir fait découvrir le monde vaste de l'écologie microbienne mais aussi pour m'avoir fait découvrir la beauté de Clermont-Ferrand: merci pour m'avoir accueilli à mon arrivée à Clermont-Ferrand et pour la visite de la ville!

J'adresse également un grand merci à Eric Peyretaillade. Merci pour tes précieux conseils, ton soutien, ton savoir et ton humour à toute épreuve. Merci pour ton aide tout au long de cette thèse. Merci pour ton investissement sur mes travaux, et sur la correction du manuscrit.

Je tiens à exprimer ma gratitude pour tous les membres de l'EA CIDAM et de l'équipe G2IM sans exception, qu'ils soient encore ici ou non, pour m'avoir accueilli chaleureusement parmi eux. Je remercie en particulier Delphine Boucher, Corinne Biderre-

Petit, Anne Moné, Brigitte Chebance (merci Brigitte pour ta grande disponibilité et ton accueil), Jean-François Brugère, Olivier Gonçalves, Sébastien Rimour, Sébastien Terrat, Ourdia Bouzid, Sophie Comtet, Nicolas Parisot, Céline Ribière, Cyrielle Gasc, William Tottey, Mohieddine Missaoui (un grand merci pour tes précieux conseils), Jérémie Denonfoux et Eric Dugat-Bony (un grand merci à vous deux pour votre aide et pour toutes nos discussions).

Merci également à mes collègues et amis du LIMOS: Luc, Jonathan C., Jonathan P. (M. Karaté! :p Je ne vais pas oublier ton calendrier Bio qui m'a accompagné durant la dernière année de ma thèse!! Merci pour ton amitié), Sébastien, Pierre, Guillaume, Romain, Nathalie, Nicolas, Toan, Rabii, Wajdi (merci pour ton amitié Bro), Sabeur, Haythem, Baraa, et tous les autres membres du LIMOS.

Je tiens notamment à remercier Béatrice Bourdieu, Pascale Gouinaud et Antoine Mahul qui ont largement contribué au bon déroulement de cette thèse.

Je n'oublie pas non plus tous mes collègues du département MMI du Puy-en-Velay pour leur accueil durant cette dernière année. Merci pour m'avoir accueilli chaleureusement parmi vous comme le membre d'une famille. Merci à vous tous pour votre soutien et votre amitié.

Ensuite, mes remerciements vont à ma famille et mes amis pour leur compréhension et leur soutien.

Cette thèse n'aurait pas abouti également sans l'aide de ma famille, en particulier, mes parents pour tout ce qu'ils ont fait pour moi, ma femme pour son soutien et pour avoir supporté ces longues années de thèse, mon petit ange Adam, mes frères et sœurs, la petite famille de ma femme et mes amis.

Enfin, je dédie cette thèse à la mémoire de mes grands-parents…

Table des matières

Table des figures

Table des tableaux

Introduction générale

Les microorganismes constituent la plus grande diversité du monde vivant et sont indispensables au bon fonctionnement des écosystèmes. En effet, ils jouent un rôle clef dans le maintien de l'équilibre des écosystèmes en participant au transfert de la matière et de l'énergie au travers des grands cycles biogéochimiques. Leurs capacités d'adaptation, leurs ont permis de coloniser tous les environnements, même les plus extrêmes. Cependant, l'exploration de cette diversité est restée longtemps très parcellaire du fait de la difficulté à cultiver les microorganismes. Ainsi, une grande majorité d'entre eux reste encore non cultivable avec les techniques actuelles.

Ces dernières années, le développement de la biologie moléculaire a révolutionné la vision de l'écologie microbienne. En effet, les techniques moléculaires développées permettent d'aborder les problématiques d'écologie microbienne via l'analyse des molécules d'acides nucléiques (Amann et al., 1995). Ces techniques indépendantes de la culture reposent sur l'utilisation de génomes entiers ou de biomarqueurs capables de renseigner sur l'identité (biomarqueurs phylogénétiques) ou sur le rôle fonctionnel (biomarqueurs fonctionnels) des microorganismes. Pour l'affiliation, les gènes exprimant la petite sous-unité de l'ARN ribosomique sont ainsi devenus des marqueurs incontournables pour décrire les communautés microbiennes (Pace, 1997). Ces gènes sont caractérisés par leur structure incluant à la fois des régions conservées facilitant leurs isolements mais aussi des régions suffisamment variables pour permettre la discrimination des microorganismes. Ils sont également ubiquistes, ne subissent pas ou peu de transfert horizontal et de recombinaisons (Hugenholtz, 2002), possèdent une taille adaptée pour les analyses et les séquences sont très abondantes dans les bases de données nucléotidiques internationales facilitant les recherches de similarité. Toutes ces caractéristiques font d'eux un marqueur moléculaire de choix largement utilisé en écologie microbienne.

Récemment, l'application du barcoding (également appelée approche amplicon) à très haut débit utilisant le séquençage de nouvelle génération (NGS pour « Next Generation Sequencing ») a permis de franchir une nouvelle étape en révélant la biosphère rare (Sogin et al., 2006). Malgré les limites de ces approches (biais de PCR, difficultés d'identification), une nouvelle vision de la diversité microbienne s'ouvre à nous. L'exploration d'échantillons complexes est alors devenue une réalité. Le séquençage systématique de tous les ADN des organismes constituant ces échantillons permet donc dans l'absolu, même si le problème est beaucoup plus complexe, l'inventaire de la totalité des gènes des communautés présentes dans un environnement donné (Lewin et al., 2014).

Les plateformes de séquençage de nouvelle génération permettent donc d'obtenir des séquences d'ADN à une vitesse sans précédent à des coûts très réduits et en font ainsi des méthodes de choix pour des études de génomique à haut débit. Cependant, en raison de la complexité de ces environnements seule une fraction des données peut être obtenue par les approches de séquençage globales et les masses de données produites sont difficiles à analyser et interpréter.

Les biopuces à ADN représentent également un outil haut débit de choix pour l'étude de milliers d'espèces en une seule expérience (Schena et al., 1995). L'évolution rapide et continue des formats de biopuces à ADN (jusqu'à plusieurs millions de spots actuellement) permet aujourd'hui d'explorer l'extrême diversité microbienne des environnements complexes. Le point clef de la réussite d'une expérience de biopuce à ADN est la détermination *in silico* de sondes à la fois spécifiques et sensibles, mais également sur la bonne interprétation des grandes quantités de données produites par cette technique. Ainsi, la complexité de la tâche de développement des biopuces est due à la difficulté de combiner une multitude de critères pour la sélection des sondes optimales (spécifique et sensible) pour des marqueurs pouvant être très similaires, ainsi qu'à l'utilisation répétitive de certains logiciels bioinformatiques, notamment ceux d'alignement de séquences qui engendrent souvent des temps de calcul prohibitifs.

L'obtention de sondes spécifiques et sensibles est un réel défi en écologie microbienne, mais la sélection de sondes exploratoires en est un encore plus grand. En effet, malgré l'explosion du nombre de séquences déposées dans les banques de données internationales, la grande majorité des séquences des microorganismes reste jusqu'alors inconnue. Pour espérer étudier la diversité microbienne dans sa globalité, il est aujourd'hui nécessaire de sélectionner des sondes exploratoires capables d'anticiper les variations génétiques et de cibler de nouvelles séquences ou variants de gènes non encore découverts. Ce besoin de sélectionner des sondes exploratoires nécessite un traitement fastidieux qui vient s'ajouter à la complexité et la grande charge de calcul nécessaire pour la sélection des sondes respectant les autres critères de qualité. Ainsi, la conception et l'analyse des biopuces à ADN exploratoires, avec leurs formats actuels, ainsi que l'explosion de l'information dans les bases de données génomiques, nécessitent un recours à « l'informatique » au sens large et plus particulièrement au calcul haute performance.

Le calcul haute performance HPC (« High Performance Computing ») est aujourd'hui au cœur de nombreuses applications dans différents secteurs comme la

recherche académique, la météorologie, la médecine, la pharmacie, la biologie, la défense, la sécurité, le transport, l'énergie, etc. En effet, grâce aux développements des nouvelles technologies et de logiciels libres, le coût du calcul intensif devient de plus en plus abordable. Cela a facilité la diffusion des équipements, des outils et des méthodes de calcul intensif qui sont ainsi devenus des outils indispensables pour répondre aux besoins des applications en termes de temps de calcul et d'espace de stockage. En biologie, l'utilisation du calcul haute performance est en évolution continue, spécialement en écologie microbienne où le traitement des données produites nécessite d'importantes capacités de calcul et de sauvegarde. Ainsi, dans ce contexte, la conception et l'analyse des biopuces à ADN exigent un recours au calcul intensif, et l'application des nouvelles approches du génie logiciel, et plus particulièrement l'ingénierie dirigée par les modèles (IDM).

L'IDM (ou « Model Driven Engineering (MDE) » en anglais), est une discipline du génie logiciel qui s'intéresse aux meilleures techniques de développement des applications informatiques et à leur évolution. C'est une ingénierie générative où l'application informatique peut être générée à partir de modèles (génération partielle ou parfois complète), et ce afin de faire face à l'évolution continue des applications et de leur complexité.

Les travaux de cette thèse s'inscrivent dans une optique d'amélioration des performances des logiciels de conception de sondes et d'analyse des biopuces à ADN pour l'écologie microbienne. Nos travaux se déclinent en plusieurs objectifs.

Notre premier objectif était d'exploiter les capacités de calcul et de stockage des architectures parallèles et distribuées afin de proposer de nouveaux algorithmes capables de sélectionner plusieurs milliers de sondes spécifiques et sensibles en un temps de calcul raisonnable.

Nous avons également souhaité optimiser et simplifier, du point de vue logiciel, les applications de sélection de sondes, et ce en faisant appel aux nouvelles approches du génie logiciel telle que l'Ingénierie Dirigée par les Modèles.

Ensuite, nous avons voulu apporter des améliorations sur une stratégie de sélection de sondes exploratoires (Militon et al., 2007), afin d'en assurer une meilleure qualité et fiabilité pour la conception de biopuces à ADN exploratoires utilisables en écologie microbienne.

Enfin, notre dernier objectif était de proposer une nouvelle méthode d'analyse des données de biopuces, pour faciliter l'interprétation des résultats issus des expériences d'hybridation des biopuces à ADN développées à l'aide des différents logiciels de sélection de sondes proposés dans le cadre de cette thèse.

Ce manuscrit aborde dans le premier chapitre le contexte général de la thèse et la problématique de la conception et de l'analyse des biopuces à ADN pour l'étude de la diversité de microorganismes dans les différents écosystèmes de notre planète. Tout d'abord, nous présentons la biodiversité microbienne et l'intérêt de son étude. Ensuite, nous exposons les différentes approches d'étude des microorganismes, de la culture aux approches moléculaires haut débit, avec un focus particulier sur l'utilisation des biopuces à ADN pour l'étude de la biodiversité. Enfin, nous évoquerons le problème de l'augmentation exponentielle des données génomiques déposées dans les banques de données publiques et son impact sur la conception des biopuces.

Le deuxième chapitre propose une synthèse bibliographique sur la sélection de sondes et l'analyse des données issues des biopuces à ADN. D'abord, nous présentons les différents critères de sélection de sondes oligonucléotidiques ainsi que les logiciels de détermination les plus couramment utilisés. Ensuite, nous exposons les différentes approches de normalisation et d'analyse des résultats des biopuces et nous présentons enfin une sélection d'outils informatiques connus pour l'analyse des biopuces.

Dans le troisième chapitre, nous proposons un état de l'art sur les méthodes et les outils informatiques permettant de répondre à notre problématique. La première partie de ce chapitre est consacrée au calcul à haute performance avec une exposition des différentes architectures parallèles et distribuées et de leur utilisation pour la parallélisation d'outils bioinformatiques en liaison avec les biopuces à ADN. La deuxième partie évoque les avancées du point de vue du génie logiciel avec un aperçu de l'ingénierie dirigée par les modèles en tant qu'approche efficace pour le développement d'applications informatiques.

Dans le quatrième chapitre, nous présentons une nouvelle approche distribuée pour la sélection de sondes exploratoires sur grilles de calcul. Nous utilisons ensuite notre logiciel pour construire une base de données complète de sondes oligonucléotidiques pour l'étude des communautés procaryotiques. Cette base de données est disponible en accès libre via un site web que nous présentons dans la deuxième partie de ce chapitre.

Dans le cinquième chapitre, nous revisitons la stratégie de sélection de sondes exploratoires utilisée dans le chapitre 4, pour une meilleure qualité de ce type de sondes. Nous implémentons notre nouvelle stratégie dans un logiciel parallèle pour la détermination de sondes oligonucléotidiques, en utilisant une approche de méta-programmation et d'ingénierie dirigée par les modèles.

Le chapitre 6 présente un nouveau logiciel pour l'analyse des données issues des expériences d'hybridation des biopuces à ADN développées à l'aide des logiciels de sélection de sondes que nous avons présentés dans les deux chapitres précédents. Notre programme utilise une approche basée sur la logique propositionnelle.

Le dernier chapitre, propose une nouvelle approche distribuée pour le calcul de la traduction inverse complète d'oligopeptides, appliquée à la sélection de sondes pour biopuces à ADN fonctionnelles. Nous utilisons une approche de méta-programmation et d'ingénierie dirigée par les modèles pour la production (par transformation) des codes sources adaptés pour différents moyens de calcul: PC, multiprocesseur de type SMP (« Symmetric Multi-Processor »), clusters ou grille de calcul.

Enfin, nous proposons une conclusion générale avec un bilan des résultats obtenus et une exposition des perspectives de recherches issues de cette thèse.

Chapitre 1: Principes des biopuces à ADN en écologie microbienne

1. Introduction

De par leur grande capacité d'adaptation et l'énorme diversité de leurs métabolismes, les microorganismes constituent la plus grande diversité du monde vivant et sont retrouvés dans tous les environnements même les plus extrêmes. Aussi, l'étude et la compréhension du fonctionnement des écosystèmes et des rôles des microorganismes représentent un enjeu majeur en écologie microbienne. Cependant, l'évaluation de l'immense diversité des microorganismes demeure très complexe (Willey et al, 2007). Dans ce chapitre, après avoir abordé la biodiversité microbienne, les différentes approches d'étude des microorganismes seront présentées avec un focus plus particulier sur les biopuces à ADN.

2. Biodiversité

2.1. Historique

La biodiversité est un terme qui désigne la diversité, la richesse et la complexité du monde vivant. Ce terme est relativement récent. Il a été proposé par Walter G. Rosen en 1985 lors des préparations du forum national sur la diversité biologique, organisé par le « National Research Council » de l'Académie des Sciences américaine en 1986. Pour des raisons d'efficacité en termes de communication, le mot biodiversité a remplacé le terme diversité biologique introduit par le biologiste américain Thomas Lovejoy en 1980. Il a été ensuite popularisé et publié pour la première fois par le biologiste américain « E.O. Wilson » à travers son livre « Biodiversity » (1988). A partir des années 80, la disparition d'espèces vivantes et la conscience du besoin de protection et de préservation de la nature ont poussé la communauté internationale à porter beaucoup plus d'intérêt à l'étude de la biodiversité. Cet intérêt a fortement augmenté dans les années 90 (Oksanen et al., 2004). L'importance de la biodiversité à cette époque est le résultat des préoccupations et des engagements exprimés pour l'environnement et le développement de notre planète.

L'étude de la biodiversité permet de lutter contre les menaces qui pèsent sur elle comme par exemple les pertes d'habitats, la pollution, le changement climatique, les invasions biologiques et l'exploitation excessive des espèces. En 2001, et suite à la demande du secrétaire général des Nations Unies Kofi Annan, un programme de travail international intitulé « l'Évaluation des écosystèmes pour le millénaire EM » a été instauré. Ce projet avait pour objectif d'évaluer les conséquences de l'évolution des écosystèmes sur l'être humain et d'établir la base scientifique des actions nécessaires pour améliorer la

restauration et l'exploitation durable des écosystèmes. Les conclusions de ce travail mené de 2001 et 2005 par plus de 1 360 experts originaires de 95 différents pays ont été présentées sous forme de cinq volumes techniques et six rapports de synthèse actuellement disponibles en plusieurs langues sur le site web du projet (http://www.unep.org/maweb/fr/index.aspx). Ces conclusions montrent une dégradation de l'environnement et de la biodiversité. En effet, selon ces rapports, environ 60 % des ressources fournies par les écosystèmes terrestres et indispensables à la vie sont surexploitées ou leur qualité a été dégradée. Le rapport conclu qu'au cours des 50 dernières années, l'être humain a causé des changements dans les écosystèmes de manière plus rapide et plus extensive que durant n'importe quelle autre période comparable de l'histoire de l'humanité. Ceci a engendré une perte substantielle et irréversible de biodiversité et une dégradation des écosystèmes. Ces changements ont réduit la capacité de la terre à répondre aux besoins humains et ont rendu le développement global et durable beaucoup plus difficile à réaliser.

Le monde vivant était auparavant classé en 5 grands règnes comprenant les animaux, les végétaux, les champignons, les protistes et les procaryotes. Les protistes regroupent les eucaryotes unicellulaires (une seule cellule). Ce groupe n'a pas de cohérence phylogénétique. Il a été initialement créé sur une base phénotypique reposant sur ce caractère de cellule unique. Ce caractère unicellulaire est également partagé par les procaryotes. La distinction entre les eucaryotes (animaux, végétaux, champignons et protistes) et les procaryotes (bactéries et archées) repose respectivement sur la présence ou l'absence d'un noyau vrai. Chez les eucaryotes les chromosomes sont contenus dans un noyau délimité par une membrane nucléaire. Chez les procaryotes, l'absence de ce compartiment nucléaire implique la présence du ou des chromosomes directement dans le cytoplasme de la cellule. Les travaux de phylogénie moléculaire de Carl Woese ont montré que les archées (Archaea) anciennement appelées archébactéries étaient distantes des bactéries et n'étaient pas aussi ancienne que précédemment supposé (Woese and Fox, 1977). Ce caractère ancien ou archaïque qui avait conduit à leur donner le nom d'archébactérie (bactérie ancestral) était supporté par la découverte de ces organismes dans des environnements extrêmes pouvant être proches de ceux de la terre primitive ayant vu l'émergence de la vie. Sur ces constatations, le monde vivant a alors été divisé non plus en 5 grands royaumes mais en 3 à savoir les bactéries, les archées et les eucaryotes.

Le nombre des espèces vivantes sur notre planète est estimé à environ 9 millions, selon les dernières estimations de la biodiversité. Cependant, la grande majorité de ces espèces ne sont pas encore décrites. En effet, seulement 1,2 million d'espèces sont répertoriées dans une base de données centrale (Mora et al., 2011). La plupart des espèces non encore décrites appartient au groupe des microorganismes.

2.2. Les microorganismes

Les microorganismes sont des organismes non visibles à l'œil nu. L'ancêtre du microscope a été développé par un mercier hollandais Antonie van Leeuwenhoek (1632-1723). Le mot « microbe » a été introduit en 1878 par le médecin militaire et chirurgien français Charles-Emmanuel Sédillot. La même année, Louis Pasteur a publié son mémoire sur « La théorie des germes et ses applications à la médecine et à la chirurgie » dans lequel il proposait la théorie de germes. En effet, Pasteur affirmait que les maladies se développent à cause de germes externes qui attaquent l'organisme, de la même manière que des microorganismes se développent dans le lait pour entraîner sa fermentation. Cette théorie a été au début refusée par les scientifiques et les médecins de l'époque qui avaient du mal à accepter qu'un microorganisme puisse tuer des organismes beaucoup plus grands. La microbiologie pasteurienne basée sur l'isolement n'a permis l'étude que d'une très faible minorité de microorganismes. En effet, la diversité des microorganismes est immense et son évaluation demeure très complexe (Willey et al., 2007).

Le nombre total de cellules bactériennes sur terre est estimé à environ $4\text{-}6 \times 10^{30}$ (Whitman et al., 1998). A titre de comparaison, le nombre d'étoiles dans l'univers est quant à lui estimé à 10^{24}. Aussi, la découverte d'un nouveau microorganisme avec de nouvelles fonctions devrait être aussi enthousiasmante que la découverte d'une nouvelle étoile (Jansson and Prosser, 2013). L'exploration des environnements complexes demeure actuellement l'un des défis majeurs en écologie microbienne du fait de l'importante diversité des microorganismes qu'ils hébergent. Pour une bonne compréhension du fonctionnement des écosystèmes il est important de:

(1) identifier les microorganismes (structure des communautés),

(2) caractériser leurs fonctions métaboliques,

(3) relier la structure à la fonction,

(3) comprendre les interactions entre microorganismes.

Un grand nombre de méthodes culturales, moléculaires et biochimiques ont été dès lors développées pour répondre à ces questions.

3. Méthodes d'étude des microorganismes

La détermination des structures et des fonctions des communautés microbiennes a nécessité le développement d'approches variées allant de la culture aux approches moléculaires haut-débit.

3.1. Approches culturale et microscopique

Les techniques classiques de microbiologie basées sur la culture consistent à inoculer un échantillon environnemental à analyser sur des milieux de culture, dans des conditions optimales pour la reproduction des populations à caractériser (température, temps d'incubation, pH, présence ou non de lumière, etc.). Ces approches sont souvent appliquées soit pour dénombrer les cellules viables (Sait et al., 2002), soit pour sélectionner des microorganismes présentant un caractère spécifique. Cependant, ces approches souffrent de certaines limites telles que la sélection préférentielle des microorganismes à croissance rapide (Sait et al., 2002) et la difficulté de reproduction des conditions naturelles de l'environnement à la base de l'isolement. De ce fait à l'heure une large proportion de microorganisme reste encore non cultivée.

Durant la dernière décennie, de nouvelles approches de culture ont été développées. Certaines utilisent des chambres de diffusion (Nichols et al., 2010) ou encore l'encapsulation de cellules dans des microgouttelettes de gel combinée à de la cytométrie en flux, pour assurer une culture à grande échelle des microorganismes (Zengler, 2002; Zengler et al., 2005).

Des approches d'observation microscopique basées sur l'utilisation des sondes fluorescentes ciblant des séquences spécifiques d'acides nucléiques, directement au niveau *in situ,* ont ensuite proposées pour apporter des données sur l'abondance, la répartition et les interactions des microorganismes d'intérêt. Les plus utilisées de ces approches sont le FISH (Fluorescent In Situ Hybridization) et le CARD-FISH (DeLong et al., 1989; Amann et al., 1990; Schönhuber et al., 1997; Bottari et al., 2006; Valm et al., 2011). De nouvelles méthodes couplant le FISH à des techniques isotopiques (MAR-FISH, SIMSISH) (Lee et al., 1999, Li et al., 2008) ont été proposées pour pouvoir accéder à la fois aux informations sur l'identité des microorganismes ainsi que sur leurs fonctions. Ces méthodes génèrent

des données quantitatives, mais ne permettent pas une étude exhaustive de tous les microorganismes d'un environnement donné. Le développement de la biologie moléculaire et l'apparition des approches haut débit se sont alors révélés comme des avancées primordiales pour appréhender cette immense diversité microbienne.

3.2. Approches moléculaires et émergence des outils haut-débit

Le développement croissant et remarquable des techniques moléculaires durant les deux dernières décennies, permet aujourd'hui de contourner les limites des approches culturales (Amann et al., 1995; Pace, 1997) et d'étudier les communautés microbiennes au travers de l'analyse des molécules d'acides nucléiques (ADN et/ou ARN). Ces techniques moléculaires reposent le plus souvent sur l'utilisation de biomarqueurs capables de fournir des informations suffisantes sur l'identité (biomarqueurs phylogénétiques) ou le rôle fonctionnel (biomarqueurs fonctionnels) d'un grand nombre de microorganismes.

Le biomarqueur phylogénétique le plus utilisé en microbiologie est le gène exprimant l'ARN ribosomique de la petite sous unité du ribosome (ARNr 16S chez les procaryotes et ARNr 18S chez les eucaryotes) (Woese et al., 1990). Les ARNr ne sont pas traduits en protéines mais exercent leurs fonctions sous forme d'ARN. Ces molécules vont s'associer aux protéines ribosomales pour former les complexes nucléprotéiques (association d'acides nucléique c'est-à-dire les ARNr et de protéines à savoir les protéines ribosomales) de la machinerie de traduction c'est-à-dire les ribosomes. Les ribosomes assurent la traduction des ARN messagers (ARNm) en protéines. Ces gènes présentent ainsi plusieurs critères qui en font de bons marqueurs phylogénétiques:

- ✓ ils sont ubiquistes (présents chez tous les organismes),
- ✓ leur taille d'environ 1500 paires de bases chez les procaryotes et 1800 chez les eucaryotes permet d'analyser un grand nombre de caractères simultanément,
- ✓ ils subissent peu ou pas de recombinaison et de transfert latéral (échanges entre espèces différentes par opposition au transfert vertical de génération en génération) ce qui assure une bonne stabilité du marqueur retraçant l'histoire évolutive,
- ✓ ils peuvent être isolés facilement grâce à la présence de régions hautement conservées (utilisation d'amorces « universelles » pour une amplification génique de ces gènes par PCR),

- ✓ ils peuvent être très discriminants grâce à la présence de régions hautement divergentes,
- ✓ les bases de données sont très riches en ces séquences facilitant les comparaisons.

Toutefois, d'autres biomarqueurs tels que les gènes codant pour la sous-unité β de l'ARN polymérase et celle de l'ADN gyrase (*gyrB*), la *recombinase A* (*recA*) ou la Protéine de choc thermique *Hsp60* ont été aussi utilisés en microbiologie pour différencier certaines espèces bactériennes (Ghebremedhin et al., 2008). En ce qui concerne les biomarqueurs fonctionnels, ils sont souvent des gènes codant pour des enzymes impliquées dans des métabolismes d'intérêt.

L'avènement du séquençage à haut débit a permis d'explorer de manière plus exhaustive la structure et les fonctions des écosystèmes complexes en dépassant les limites des approches d'empreintes moléculaires moins informatives (DGGE, TGGE, ARISA, ARDRA, SSCP, T-RFLP). Ainsi, la première étude métagénomique a exploré les communautés microbiennes de la mer des Sargasses (Venter, 2004). Sur la base de millions de séquences, de nouvelles branches du monde microbien ont été découvertes ainsi que de nouvelles fonctions métaboliques. Une seule étude met ainsi en évidence des millions de nouveaux gènes appartenant pour certains à de nouvelles espèces. Ces séquences enrichissent les bases de données qui croissent toujours de façon exponentielle.

Les techniques de séquençage de nouvelle génération permettent de s'affranchir de l'étape de clonage des fragments d'ADN, de réduire considérablement les coûts et le temps d'acquisition des données, et d'augmenter les quantités de données produites (Shendure and Ji, 2008; Ansorge, 2009; Metzker, 2010). Ces progrès de séquençage de deuxième génération ont permis d'aborder de nombreux projets de métagénomique (aujourd'hui 502 études référencées dans la base de données « Genome OnLine Database » (GOLD) (www.genomesonline.org, consulté le 06 Mars 2014). Cependant, en raison de la taille relativement courte des fragments séquencés (moins de 700 paires de bases (bp)), leur assemblage pour assurer la reconstruction de la séquence complète de génomes pouvant faire plusieurs Mpb reste problématique (Morales and Holben, 2011). De plus, la quantité de données générées reste encore insuffisante pour accéder aux génomes de l'ensemble des microorganismes présents dans un environnement complexe. En effet, selon Quince et al. (2008), les capacités de séquençage actuelles devraient être multipliées par plus de 10^4 fois pour couvrir 90% de la diversité de certains écosystèmes particulièrement complexes. Il

faudrait ainsi mener plusieurs milliers d'expérimentations (« run ») de séquençage ce qui reste inenvisageable que ce soit sur le plan technique ou financier (Quince et al., 2008).

Plus récemment, une troisième génération, se basant sur les progrès des nanotechnologies, a été proposée pour améliorer encore les données générées (Blow, 2008; Munroe and Harris, 2010; Pareek et al., 2011). Ces approches en cours de développement devraient à terme permettre la lecture directe d'une seule molécule d'ADN sur une longueur de plusieurs milliers de bases.

Le séquençage est donc en constante évolution mais génère des quantités de données qui ne sont pas toujours facilement exploitables. Dans ce contexte, l'utilisation, à l'échelle du laboratoire, d'une autre approche moléculaire alternative, complémentaire et aussi de haut débit peut alors être privilégiée, il s'agit des biopuces à ADN.

4. Biopuces à ADN

4.1. Principe des biopuces à ADN

La technologie des biopuces à ADN (ou « DNA microarrays » en anglais) a été initialement mise au point au milieu des années 90, suite au séquençage des premiers génomes, afin d'étudier simultanément l'expression de tous les gènes d'un organisme (Schena, et al., 1995). Son concept repose sur un procédé multidisciplinaire qui intègre la biologie, la chimie des acides nucléiques, les nanotechnologies, l'analyse d'images et la bioinformatique.

Une biopuce à ADN est une technologie miniaturisée de biologie moléculaire constitué d'un support solide, souvent une lame de verre, sur lequel sont fixés jusqu'à plusieurs millions de fragments ADN simple brin appelés sondes. Les positions des sondes sur la biopuce sont connues. Le principe des biopuces (figure 1) est alors basé sur la capacité de reconnaissance et d'hybridation entre les sondes (ADNg, produits PCR, ADNc ou oligodésoxyribonucléotides) et leurs cibles simple brin (produits PCR, ADNg ou ARN) dans l'échantillon à analyser.

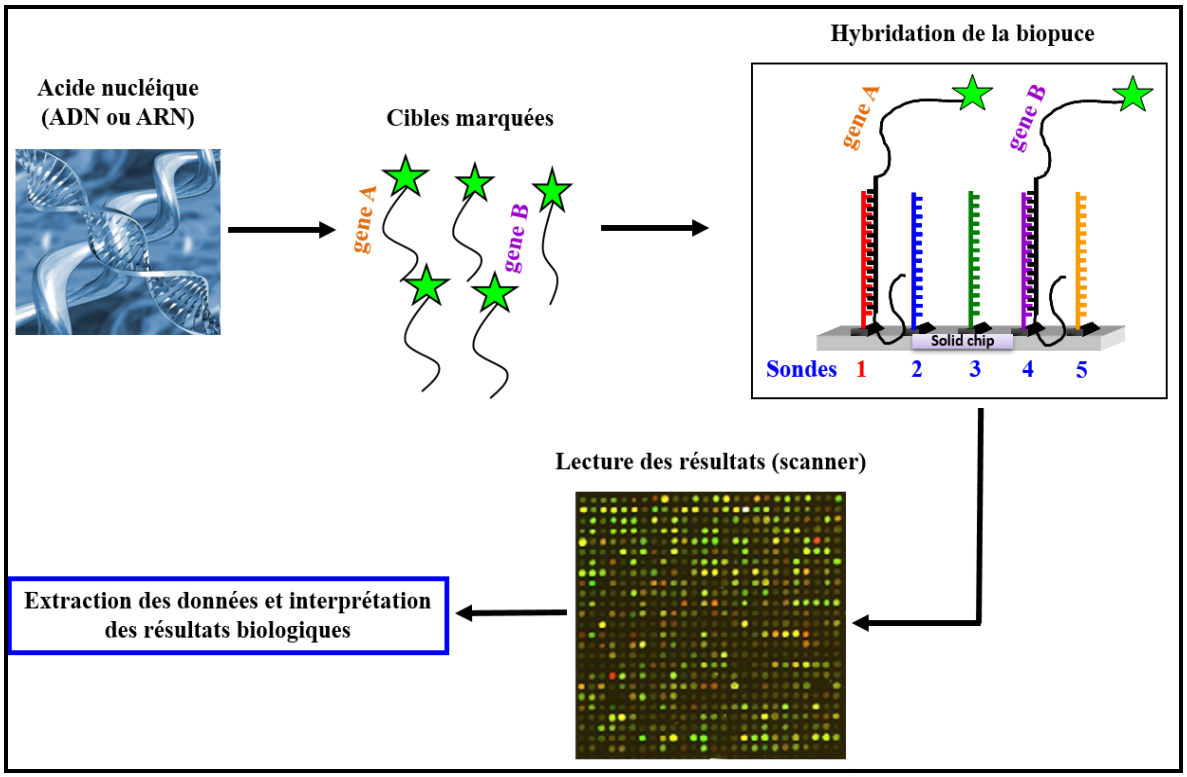

Figure 1. Principe des biopuces à ADN. Les sondes spécifiques fixées sur le support solide de la biopuce s'hybrident spécifiquement avec les cibles par complémentarité des bases. Les cibles extraites des échantillons biologiques à analyser sont marquées à l'aide d'un fluorochrome. Après, l'étape d'hybridation des cibles sur la biopuce et la lecture de la biopuce avec un scanner, l'image obtenue est alors analysée permettant les interprétations biologiques.

4.2. Formats des biopuces à ADN et utilisations en écologie microbienne

Initialement, les sondes désoxyribonuléotidiques étaient fixées sur des membranes de nylon ou de nitrocellulose et les cibles étaient marquées par de la radioactivité (P^{32}). En raison de la densité très limitée en nombre de sondes, ce support a été remplacé par un format lame de verre pré activée par une chimie de surface permettant la fixation d'un nombre beaucoup plus élevé de sondes. Les cibles radioactives, quant à elles, ont été remplacées par des cibles marquées par des composés fluorescents en raison de leur flexibilité d'utilisation et de leur facilité de manipulation. Ces nouveaux formats de biopuce de la taille d'une lame de microscope sont actuellement les plus utilisés grâce à leur haute densité, la possibilité de fixation covalente des sondes sur le support solide, la

diminution du bruit de fond et la possibilité d'analyser au moins deux échantillons de cibles biologiques lors d'une même expérimentation (Dharmadi and Gonzalez, 2004).

Les biopuces à ADN *in situ* sont actuellement les biopuces les plus utilisées. La synthèse des sondes est réalisée directement sur le support solide de la biopuce. Ce type de biopuce permet une flexibilité de synthèse des sondes oligonucléotidiques avec une très haute densité. La société Nimblegen (http://www.nimblegen.com) avait développé des biopuces à très haut débit comportant jusqu'à plus de 4 millions de sondes sur un format de type lame de microscope. Notons cependant que les biopuces Nimblegen ne sont plus commercialisées depuis 2012. La société Agilent (http://www.home.agilent.com) propose quant à elle des formats de biopuces à très haut débit pouvant comporter sur une même biopuce plusieurs réseaux de sondes autorisant l'étude de 24 échantillons en même temps.

Ces biopuces à très haut débit sont donc des outils de choix pour aborder les différentes questions que posent l'écologie microbienne (Cook and Sayler, 2003; Ehrenreich, 2006; Gentry et al., 2006; Loy and Bodrossy, 2006; Sessitsch et al., 2006; Wagner et al., 2007) notamment sur la structure des génomes, leurs parentés, les expressions géniques, les capacités métaboliques, la structure et la dynamique des communautés microbiennes (figure 2).

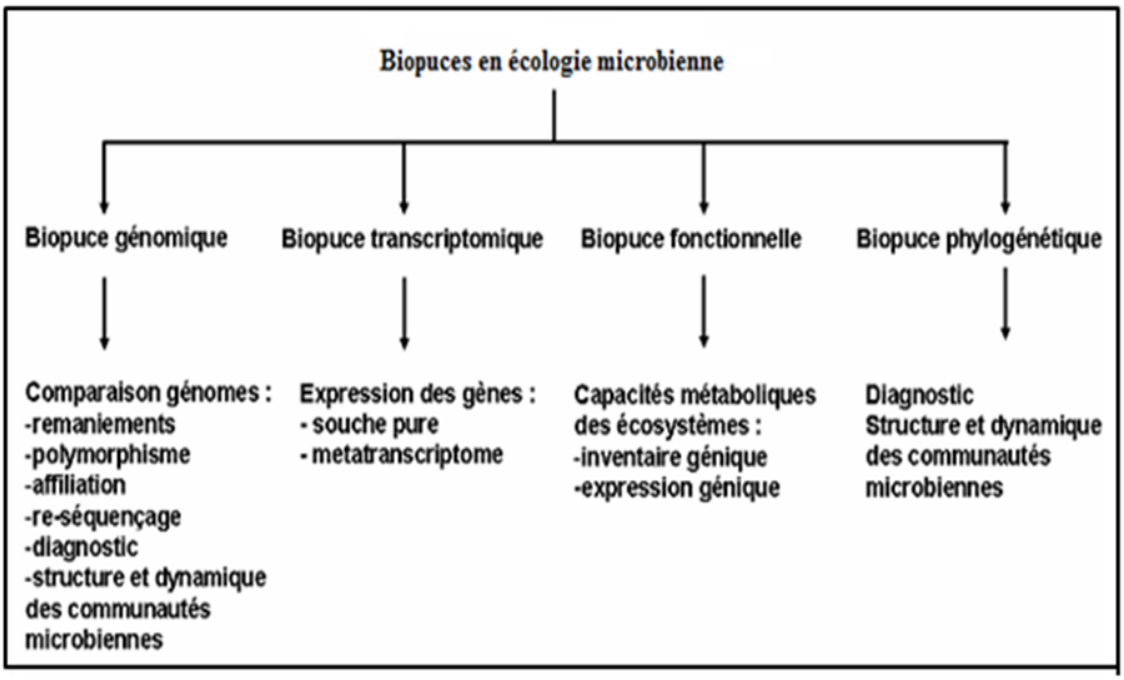

Figure 2. Présentation des différentes applications des biopuces à ADN en écologie microbienne.

La première biopuce à ADN appliquée à l'écologie microbienne a été développée en 1997 (Guschin et al., 1997). Elle était conçue pour l'identification des bactéries nitrifiantes. Cette biopuce contenait 9 sondes ciblant le gène exprimant l'ARN ribosomique 16S. Depuis, les biopuces à ADN ont été largement utilisées en écologie microbienne (Wagner et al., 2007; Zhou, 2003).

Les biopuces phylogénétiques sont actuellement largement utilisées pour étudier la structure et la dynamique des communautés microbiennes.

Plusieurs biopuces phylogénétiques ont été développées pour explorer la diversité microbienne de différents environnements tels que les écosystèmes lacustres, les estuaires, les sols, les boues activées, les aérosols urbains. Ces biopuces ont été utilisées pour:

(1) étudier la diversité totale des environnements complexes (Wilson et al., 2002; DeSantis et al., 2005; Brodie et al., 2006; Palmer et al., 2006; Militon et al., 2007; Huyghe et al., 2008);

(2) cibler des groupes microbiens fonctionnels comme les sulfato-réducteurs (Loy et al., 2002), les nitrifiants (Kelly et al., 2005), les acidophiles (Yin et al., 2007; Garrido et al., 2008);

(3) caractériser des groupes taxonomiques comme les béta-protéo-bactéries de l'ordre des Rhodocyclales (Loy et al., 2005), les entérobactéries (Lehner et al., 2005), les alpha-protéo-bactéries (Sanguin et al., 2006), les *Pseudomonas* (Sanguin et al., 2008), les cyanobactéries (Castiglioni et al., 2004).

Notons que la PhyloChip **(**Brodie et al., 2006**)** a été la première biopuce à ADN phylogénétique capable de caractériser les structures microbiennes en ciblant de manière exhaustive le marqueur phylogénétique 16S**.** Cette biopuce est composée actuellement de 500.000 sondes permettant de cibler 9000 unités taxonomiques opérationnelles. Elle a été appliquée pour caractériser les communautés procaryotes des écosystèmes tels que l'atmosphère urbaine (Brodie et al., 2007), les sols de prairie (Cruz-Martinez et al., 2009; DeAngelis et al., 2009), les sols de l'Antarctique (Yergeau et al., 2009), les sols miniers (Rastogi et al., 2010a, 2010b), les sédiments fluviaux contaminés par des métaux (Rastogi et al., 2011), les feuille d'agrumes (Sagaram et al., 2009), les aspirations endotrachéales chez des patients colonisés par *Pseudomonas aeruginosa* (Flanagan et al., 2007), etc.

La biopuce fonctionnelle GeoChip 4.0 (Tu et al., 2014) est la biopuce généraliste la plus aboutie. Elle est composée de 83 992 sondes oligonucléotides de 50-mers[1] ciblant 152 414 gènes dans 410 catégories géniques pour différentes fonctions et processus biochimiques dont les cycles du carbone, de l'azote, du soufre et du phosphore, les processus énergétiques, la résistance et la réduction des métaux, la dégradation des contaminants organiques, les réponses au stress, la résistance aux antibiotiques et les bactériophages (Rhee et al., 2004; He et al., 2007, 2010, 2011). L'évolution de la GeoChip se poursuit avec une nouvelle version 5.0, récemment développée, composée de 167 044 sondes couvrant 395 894 séquences codantes (CDS) d'environ 1 500 familles de gènes fonctionnels. D'autres biopuces fonctionnelles explorant des voies métaboliques plus ciblées ont également été développées comme par exemple les biopuces ciblant les voies de dégradation des hydrocarbures aromatiques polycycliques (Terrat et al., 2010) ou les voies de dégradation des solvants chlorés (Dugat-Bony et al., 2012a).

La pertinence notamment des biopuces phylogénétiques en écologie microbienne ainsi que leur complémentarité aux nouvelles techniques de séquençage ont été mises en évidence par différentes études. Ces études ont montré une forte corrélation entre les résultats des biopuces et des NGS (Ahmed et al., 2009, Ahn et al., 2011, Tottey et al., 2013).

L'une des difficultés majeures de l'approche biopuces à ADN, réside dans la détermination de sondes spécifiques et sensibles (pour revue voir Dugat-Bony et al., 2012b).

4.3. Détermination de sondes pour biopuces à ADN

Une alternative aux sondes ADN simple brins est l'utilisation de sondes oligonucléotidiques. Ces sondes sont faciles à synthétiser (Kreil et al., 2006). Elles possèdent une taille plus petite (généralement entre 20 et 70-mers) qui permet d'augmenter leur spécificité mais de diminuer cependant leur sensibilité. La spécificité de ces sondes doit cependant être évaluée pour savoir si elles peuvent potentiellement entraîner des hybridations aspécifiques avec des séquences non ciblées. Dans ce contexte, une analyse bioinformatique est nécessaire pour choisir les sondes les plus efficaces. En effet, la

[1] La longueur des oligonucléotides est souvent donnée en utilisant le mot « mer » (du grec « meros »: partie). Par exemple, un oligonucléotide de 50-mers désigne un fragment ADN de 50 nucléotides.

caractérisation des communautés microbiennes, par une approche biopuces à ADN, repose sur l'efficacité des sondes sélectionnées. Pour sélectionner des sondes performantes, il est nécessaire de se baser sur les critères prédisant l'efficacité de l'hybridation sondes/cibles (Pozhitkov et al., 2007). Ainsi, plusieurs critères interviennent dans le choix des sondes oligonucléotidiques:

- ✓ une sonde doit s'hybrider parfaitement avec sa cible. Elle doit être facilement accessible et ne doit pas former des structures secondaires,
- ✓ une sonde doit s'hybrider uniquement avec sa cible, les hybridations croisées doivent être évitées (Kane et al., 2000),
- ✓ toutes les sondes sélectionnées doivent posséder un comportement thermodynamique uniforme.

La sélection de sondes oligonucléotidiques exige une connaissance préalable des séquences des gènes ciblés mais également des séquences non ciblés présentent au sein de l'échantillon étudié. Elle nécessite alors d'exploiter les données de séquences existantes dans les bases de données génomiques.

C'est au début des années 80 que les premières banques de séquences sont apparues. Pour les séquences protéiques, deux banques principales ont été initialement créées. Il s'agit de « Protein Information Resource » (PIR) (George et al., 1986; Wu et al., 2003) et Swissprot (Bairoch and Boeckmann, 1993, 1994; Bairoch and Apweiler, 2000).

D'autres banques de séquences protéiques ont été ensuite développées comme Genpept et TrEMBL. En 2002, une collaboration entre PIR, EBI (« European Bioinformatics Institute ») et SIB (« Swiss Institute of Bioinformatics »), financés par les «National Institutes of Health, USA » (NIH), a permis la mise en place d'une unique base de données mondiale de séquences protéiques, en fusionnant les bases de données de PIR, Swiss-Prot, et TrEMBL.

En ce qui concerne les séquences des acides nucléiques, il existe actuellement 3 principales banques de données généralistes répertoriant ces séquences ainsi que leurs descriptions détaillées (fiches descriptives): DDBJ (DNA DataBank of Japan), GenBank (National Institue of Health genetic sequence database) et EMBL-Bank (European Molecular Biology Laboratory).

A titre d'exemple, les séquences stockées dans EMBL-Bank sont décrites suivant des classes de données préétablies. Ces classes donnent entre autres l'origine des séquences et leur méthode d'obtention. Les séquences sont également organisées et classées dans des sous divisions (tableau 1). La EMBL-Bank, ainsi que deux autres bases de données à savoir « Sequence Read Archive » (SRA) et « Trace Archive », gérées elles aussi par l'institut européen de bioinformatique, constituent ensemble la banque ENA (« European Nucleotide Archive ») (Leinonen et al., 2011). ENA stocke également des informations complémentaires telles que les procédures expérimentales, les détails de l'assemblage des séquences ainsi que d'autres données liées aux projets de séquençage (http://www.ebi.ac.uk/ena/).

Tableau 1: Les divisions des données dans EMBL.

(Source: http://www.ebi.ac.uk/embl/Documentation/User_manual/usrman.html, mai 2013)

Division	Code
Bacteriophage	PHG
Environmental Sample	ENV
Fungal	FUN
Human	HUM
Invertebrate	INV
Other Mammal	MAM
Other Vertebrate	VRT
Mus musculus	MUS
Plant	PLN
Prokaryote	PRO
Other Rodent	ROD
Synthetic	SYN
Transgenic	TGN
Unclassified	UNC
Viral	VRL

Les données de ces trois principales banques de données nucléiques (DDBJ, GenBank et ENA) sont décrites et organisées suivant une charte internationale permettant une compréhension universelle de la description et de l'annotation des séquences

disponibles. En effet, un programme de collaboration internationale entre GenBank et EMBL, a vu le jour en février 1986. Cette collaboration, aujourd'hui appelée « International Nucleotide Sequence Database Collaboration » (INSDC), s'est ensuite étendue avec la participation de DDBJ en 1987 (figure 3). Elle a donné naissance en 1990 à un dispositif de normes communes pour la pratique d'annotation de séquences. Un format unique et universel pour la description des caractéristiques biologiques des séquences déposées dans les banques de données nucléiques a été alors mis en place (« DDBJ/EMBL/GenBank Feature Table Definition »). Dans le cadre de cette collaboration, les trois banques échangent régulièrement les séquences collectées. Depuis, elles hébergent et partagent pratiquement les mêmes données. Ces données ont connu une augmentation quasi-exponentielle (figure 4) surtout avec les quantités de séquences générées par les approches de séquençage à haut débit.

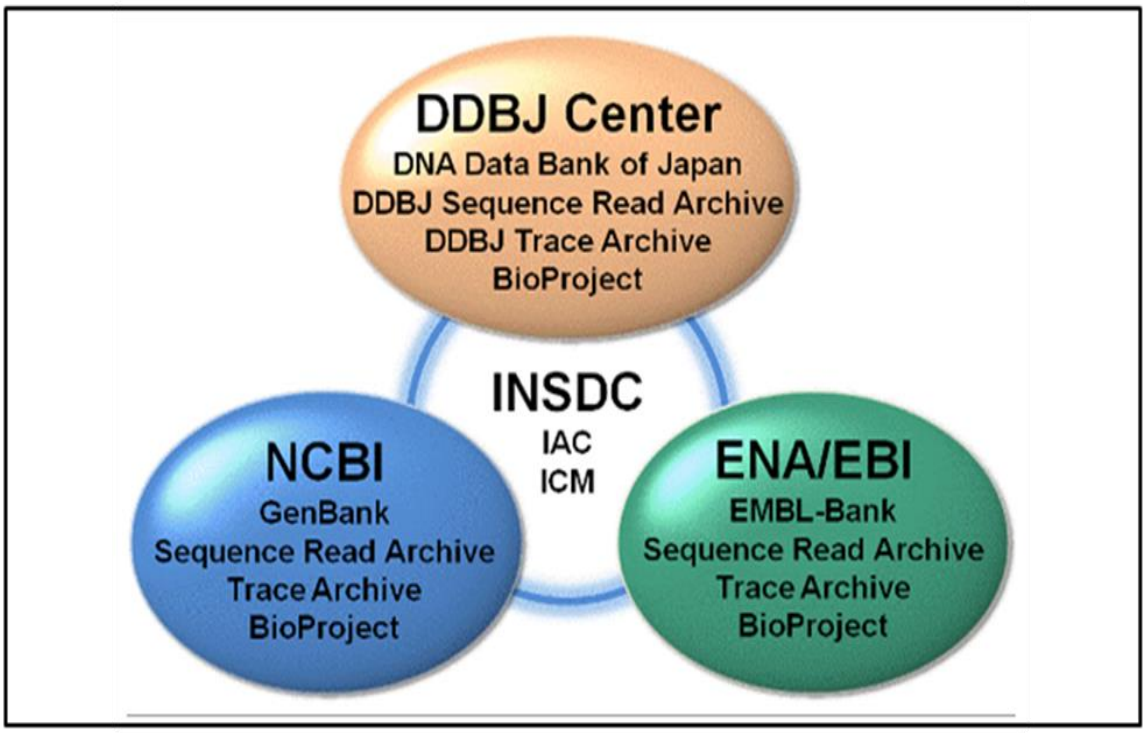

Figure 3. Schéma de la collaboration entre EMBL-Bank, GenBank et DDBJ: « International Nucleotide Sequence Database Collaboration » (INSDC) (Source: http://www.ddbj.nig.ac.jp/intro-e.html, mai 2013).

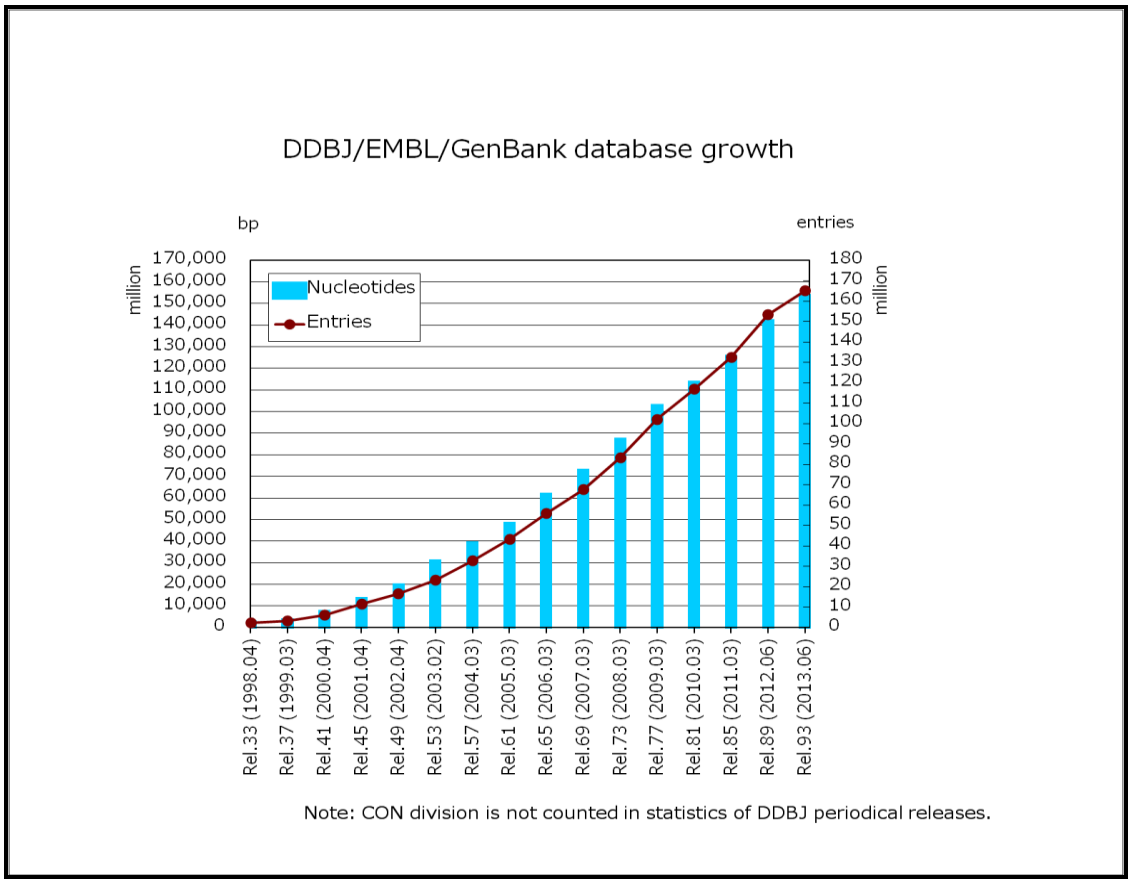

Figure 4. Evolution des quantités de données nucléotidiques déposées dans la base de données EMBL-Bank/GeneBank/DDBJ entre 1998 et 2013 (Source: http://www.ddbj.nig.ac.jp/breakdown_stats/dbgrowth-e.html#dbgrowth-graph, aout 2013).

L'explosion des données et l'hétérogénéité des séquences contenues dans les principales banques de séquences généralistes, ont encouragé la création d'autres bases de données spécialisées. Ces bases sont généralement plus homogènes. Elles sont développées autour d'une thématique bien précise ou pour réunir les séquences d'un même organisme, ou également pour lever des ambiguïtés des banques publiques généralistes et corriger des éventuels problèmes d'annotation, de classification voir de la séquence elle-même. Parmi ces banques, on peut citer les banques spécialisées des gènes codant la petite sous unité de l'ARNr comme Greengenes (DeSantis et al., 2006), SILVA (Quast et al., 2013), RDP (Cole et al., 2009) qui sont très utilisées en écologie microbienne.

Les outils de sélection de sondes doivent donc surmonter plusieurs défis pour assurer une utilisation à haut débit à la fois efficace et fiable des biopuces à ADN. En effet, ces outils doivent traiter les immenses quantités de données de séquences pour explorer la

diversité complète des communautés microbiennes. Ils doivent également prendre en compte la nature de ces données, qui restent incomplètes et contiennent souvent des erreurs de séquençage d'annotation ou d'affiliation.

4.4. Analyse des résultats de biopuces à ADN

La dernière étape d'une expérience de biopuce à ADN correspond à la mesure des résultats de l'hybridation à l'aide d'un scanner approprié. Cette dernière produit une image numérique représentant la surface de la biopuce et les niveaux de signal d'hybridation sonde/cible produit pour chaque spot[2]. Dans une expérience de biopuce classique à deux couleurs (dont chacune correspond à un fluorochrome), deux images monochromes, généralement en format TIFF (« Tagged Image File Format »), sont produites: une pour chaque longueur d'onde. Elles sont ensuite superposées *in silico* pour fournir une image allant du vert au rouge et représentant les deux conditions d'hybridation (figure 5). Les images obtenues représentent le reflet qualitatif et quantitatif des hybridations.

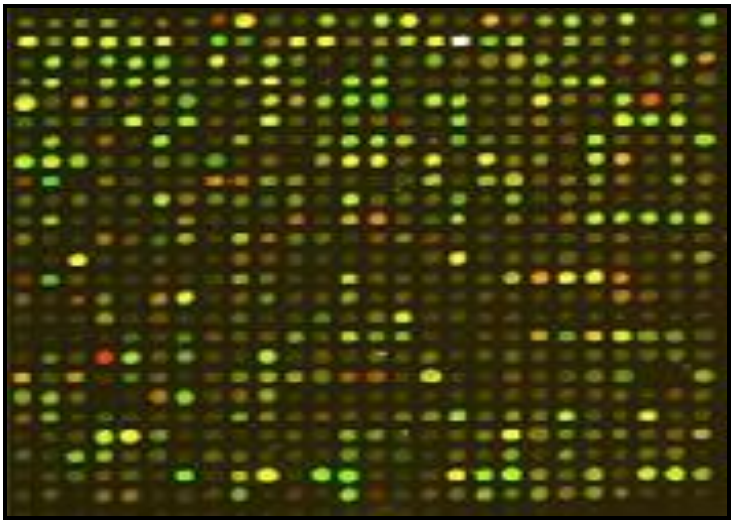

Figure 5. Exemple d'image d'une biopuce à ADN produite par un scanner: les couleurs reflètent les niveaux d'intensité de fluorescence permettant la quantification des hybridations sonde/cible.

L'analyse des images produites par le scanner, pour l'interprétation des résultats d'une expérience de biopuce, consiste en plusieurs traitements successifs (Ehrenreich,

[2] Un spot est un point d'hybridation situé sur le support solide de la biopuce, dans lequel sont fixées les sondes.

2006): l'extraction des données numériques, la normalisation des données et enfin l'analyse et l'interprétation biologique de ces données.

4.4.1. Extraction des données numériques à partir des images de biopuces

C'est la première phase du processus d'analyse des données de biopuces. Cette étape a un fort impact sur la qualité des résultats pouvant être obtenus au final. L'objectif est de fournir, à partir de l'image de biopuce, une mesure de l'intensité d'hybridation sonde/cible pour chaque spot.

L'image produite par le scanner est analysée par un logiciel d'analyse d'images de biopuces à ADN. Ce logiciel est souvent fourni avec le scanner et dépend du type et de la marque de la biopuce utilisée.

L'analyse de l'image est réalisée en trois étapes:

(1) **L'adressage:** c'est l'identification de la position des spots sur la lame. On parle souvent d'appliquer une grille sur l'image. Dans un premier temps, la grille est positionnée approximativement en fonction des paramètres du plan de dépôt: nombre de blocs, nombre de lignes et de colonnes d'un bloc, espacement entre les blocs et entre les spots à l'intérieur d'un bloc, nombre de spots par bloc, etc. Un ajustement de la grille est ensuite réalisé afin de corriger les biais éventuels du processus mécanique de fixation de sondes.

(2) **La segmentation des spots:** c'est l'identification des pixels appartenant au spot et qui seront utilisés pour la mesure de l'intensité des sondes, ainsi que les pixels appartenant au fond et qui seront utilisés pour la mesure du bruit de fond.

(3) **Le calcul des valeurs numériques:** c'est le calcul des valeurs d'intensité pour chaque spot. Ces valeurs sont calculées pour chaque condition d'hybridation identifiée par l'un des fluorochromes utilisés. Les valeurs du bruit de fond peuvent aussi être calculées si le logiciel le permet. Les spots non exploitables à cause des biais de l'hybridation sont repérés pour ne pas être considérés lors de l'analyse des données. D'autres valeurs peuvent également être calculées comme les logarithmiques et les ratios des intensités.

4.4.2. Normalisation des données numériques

En raison de la complexité de mise en œuvre d'une expérience d'hybridation de biopuce à ADN, de nombreux biais techniques potentiels peuvent affecter de manière non négligeable la mesure des intensités des fluorescents. Ces biais sont parfois inévitables car certains ne peuvent pas être parfaitement contrôlables tels que la différence entre les rendements des fluorochromes Cy3 et Cy5 ou la quantité de sondes déposées dans les spots. Une discussion sur les sources de ces biais techniques peut être trouvée dans (Hartemink et al., 2001). La normalisation des données issues des biopuces peut corriger les effets possibles de ces biais et garantir la distinction des variations biologiques que nous voulons mettre en évidence, des variations causées par des biais techniques. La normalisation se base sur certaines hypothèses, en particulier, celle qui suppose que pour une expérience d'hybridation de biopuce, seulement un nombre relativement faible de séquences ciblées varie entre les différents échantillons hybridés. La validité de cette hypothèse est principalement liée au nombre de séquences différentes présentes dans les échantillons et ciblés par la biopuce. En effet, dans le cas de l'étude d'un grand nombre de séquences, cette l'hypothèse peut être considérée comme réaliste et ainsi la normalisation peut être effectuée en utilisant l'ensemble total des sondes. Pour un nombre moyen de séquences, l'hypothèse devient moins réaliste et la normalisation peut dans ce cas être réalisée en utilisant un sous ensemble de séquences déterminées *a priori* (par exemple séquences correspondant à des gènes de ménage ou cibles de contrôle) et/ou *a posteriori* (gènes invariants). Cependant, si le nombre de séquences étudiées est faible, la normalisation ne doit être réalisée, de préférence, qu'en se basant sur les intensités des sondes de contrôle utilisées sur la biopuce. Après avoir déterminé les ensembles de sondes sur lesquelles la normalisation doit se baser, il est important de bien choisir la méthode de normalisation adéquate à appliquer. Certaines méthodes de normalisation peuvent être utilisées ensemble d'une manière successive.

4.4.3. Analyse et interprétation biologique des résultats

C'est la dernière étape d'une expérience de biopuce à ADN. Elle consiste à interpréter plusieurs milliers de valeurs numériques afin d'en extraire une information biologique pertinente. Cette étape est certainement la plus difficile du processus d'analyse des images de biopuces. Les objectifs de l'interprétation biologique des résultats peuvent largement varier en fonction du type de la biopuce, de la plateforme utilisée, de l'application et de la nature des échantillons hybridés. L'analyse des données peut ainsi

consister à déterminer la composition de l'échantillon hybridé, identifier les gènes sur ou sous-exprimés d'un échantillon à un autre, regrouper les gènes ayant le même profil d'expression, trouver les gènes expliquant les variations d'expression d'autres gènes, etc.

5. Conclusion

Nous avons vu dans ce chapitre que l'étude de l'immense diversité des microorganismes, exige actuellement l'utilisation des approches génomiques à haut débit telle que les biopuces à ADN. Cependant, et malgré les efforts continus de la communauté scientifique et le grand nombre de séquences déposées régulièrement dans les banques de données génomiques, la grande majorité des microorganismes reste encore inconnue. L'étude des communautés microbiennes nécessite donc le développement de nouvelles approches permettant à la fois de traiter l'immense quantité de données génomiques disponibles et d'explorer la grande diversité des microorganismes non encore découverts. L'utilisation de la technologie de biopuces à ADN combinée avec une nouvelle approche de sélection de sondes spécifiques et exploratoires représente donc une piste intéressante pour relever un tel challenge. Cependant, le développement de ces sondes exploratoires nécessite de nouveaux outils et solutions informatiques capables de traiter une grande quantité de données et de gérer une complexité de plus en plus importante en fonction de la nature de la biopuce développée. L'obtention de ces outils nécessite donc de faire appel aux concepts récents du génie logiciel (Ingénierie Dirigée par les Modèles) et aux techniques du calcul intensif. Ces concepts et techniques sont décrits dans le chapitre suivant, après une discussion sur les différentes approches de sélection de sondes et d'analyse des données issues des biopuces.

Chapitre 2: Etat de l'art: Approches existantes pour la conception et l'analyse des biopuces à ADN

1. Introduction

De nombreuses méthodes de conception et d'analyse des biopuces ont été développées. Dans ce chapitre, après avoir décrit les principaux critères de sélection de sondes pour biopuces à ADN, un aperçu détaillé des principales stratégies de détermination de sondes actuellement disponibles pour construire des biopuces phylogénétiques et/ou fonctionnelles sera présenté avec un focus particulier sur la sélection de sondes exploratoires. Ensuite, nous expliquerons la démarche nécessaire pour l'analyse des résultats d'une expérience de biopuce à ADN ainsi que les principales méthodes utilisées.

2. Sélection de sondes pour biopuces à ADN en écologie microbienne

L'étape cruciale pour le développement des biopuces à oligonucléotides dédiées à l'écologie microbienne, est la détermination des sondes. Bien que de nombreux outils généraux de détermination de sondes soient actuellement librement accessibles (Lemoine et al., 2009, Dugat-Bony et al., 2012b), seuls quelques-uns peuvent être appliqués en écologie microbienne (tableau 2). Dans les paragraphes suivants, nous allons définir les différents principes et critères de détermination de sondes. Nous présenterons ensuite les principaux outils de détermination de sondes oligonucléotidiques pour l'écologie microbienne, en particulier ceux qui combinent l'utilisation des sondes dégénérées et non-dégénérées. Nous montrerons ainsi comment certaines stratégies de détermination de sondes ont essayé de contourner le problème de disponibilité des séquences dans les banques de données publiques, pour rendre possible l'étude de nouveaux microorganismes pour lesquels aucune séquence n'est disponible.

2.1. Critères de sélection de sondes oligonucléotidiques

La sélection de sondes doit prendre en compte plusieurs critères qui sont:

- ✓ la spécificité (les sondes ne doivent pas s'hybrider avec des séquences non cibles),
- ✓ la sensibilité (les sondes doivent détecter des cibles en faible abondance dans des mélanges complexes),
- ✓ l'uniformité (les sondes doivent avoir un comportement d'hybridation similaire) (Loy and Bodrossy, 2006; Wagner et al., 2007).

Les programmes de détermination de sondes, disponibles actuellement, diffèrent dans le choix des critères considérés pour choisir le meilleur ensemble de sondes (tableau 3).

2.1.1. La spécificité

La spécificité des sondes est connue comme étant le critère de sélection de sondes le plus important pour garantir la qualité des résultats d'hybridation de biopuces (Kane et al., 2000; Evertsz et al., 2001; Koltai and Weingarten-Baror, 2008). En effet, lors de la phase d'hybridation, chaque sonde oligonucléotidique est mise en contact avec le mélange cible qui peut contenir un très grand nombre de molécules d'acides nucléiques différentes. Il est donc nécessaire qu'une sonde donnée s'hybride uniquement avec sa séquence cible, et non avec d'autres séquences présentes dans le mélange. Dans ce cas seulement la sonde est considérée comme spécifique de sa cible. Pour prendre en compte ce critère il faut rechercher les similarités potentielles entre la sonde et les séquences non ciblées. Les hybridations non spécifiques (ou croisées) engendrées par ses appariements non souhaités peuvent fausser l'interprétation des résultats par l'augmentation de l'intensité des signaux d'hybridation (figure 6).

Tableau2. Logiciels de sélection de sondes dédiés à l'écologie microbienne. Abréviations: POA « Phylogenetic Oligonucleotides Array », FGA « Functional Gene Array », WGA « Whole Genome Array ».

Logiciel	Applications	Disponibilité	URL	Référence
ARB	POA	Téléchargeable, Linux, MacOS	www.arb-home.de/	Ludwig et al., 2004
PRIMROSE	POA	Téléchargeable, Linux, Windows, MacOS	-	Ashelford et al., 2002
ORMA	POA, FGA	Téléchargeable sur demande	-	Severgnini et al., 2009
PhylArray	POA	Web Interface	g2im.u-clermont1.fr/serimour/phylarray/	Militon et al., 2007
HPD	FGA	Téléchargeable, Windows	brcapp.kribb.re.kr/HPD/	Chung et al., 2005
ProDesign	FGA	Interface Web	uhnresearch.ca/labs/tillier/ProDesign/ProDesign.html	Feng and Tillier, 2007
HiSpOD	FGA, WGA	Interface Web	fc.isima.fr/~g2im/hispod/	Dugat-Bony et al., 2011
Metabolic Design	FGA	Téléchargeable sur demande, Windows	-	Terrat et al., 2010
CommOligo 2.0	FGA, WGA	Téléchargeable, Windows	ieg.ou.edu/software.htm	Li et al., 2005
OligoWiz 2.0	FGA, WGA	Téléchargeable, Linux, Windows, MacOS	www.cbs.dtu.dk/services/OligoWiz2/	Wernersson and Nielsen, 2005
ROSO	FGA, WGA	Interface Web	pbil.univ-lyon1.fr/roso/Home.php	Reymond et al., 2004
ArrayOligoSelector	FGA, WGA	Téléchargeable, Linux	arrayoligosel.sourceforge.net/	Bozdech et al., 2003
OligoArray 2.1	FGA, WGA	Téléchargeable, Linux	berry.engin.umich.edu/oligoarray2_1/	Rouillard et al., 2003
OligoPicker	FGA, WGA	Téléchargeable, Linux	pga.mgh.harvard.edu/oligopicker/	Wang and Seed, 2003
PROBEmer	POA, FGA, WGA	Interface Web	-	Emrich et al., 2003
YODA	FGA, WGA	Téléchargeable, Linux, Windows, MacOS	-	Nordberg, 2005
ProbeSelect	WGA	Téléchargeable sur demande, Linux	-	Li and Stormo, 2001
KASpOD	POA, FGA	Interface Web	g2im.u-clermont1.fr/kaspod/	Parisot et al., 2012

Tableau 3. Critères de sélection des sondes considérés selon les logiciels.

Logiciel	Taille sondes	Structure secondaire	Faible Complexité	% G+C	Tm	ΔG	Sondes dégénérées	Specificité	Base de données
ARB	Choisi par l'utilisateur (10-100)	Non	Non	Oui	Oui	Non	Non	Alignement local + calcul thermodynamique	ARB Silva database
PRIMROSE	Choisi par l'utilisateur (3-100)	Non	Non	Non	Non	Non	Oui	-	-
ORMA	Choisi par l'utilisateur	Non	Oui	Non	Oui	Non	Oui	Non	Non
PhylArray	Choisi par l'utilisateur (20-70)	Non	Non	Oui	Oui	Non	Oui	Blast + critères de Kane	Base de données ARNr personnalisée.
HPD	Choisi par l'utilisateur (20-70)	Oui	Non	Oui	Oui	Oui	Non	Blast + critères de Kane	Ensemble de données en entrée.
ProDesign	Choisi par l'utilisateur (20-70)	Oui	Oui	Oui	Oui	Oui	Non	Suffix Tree + critères de Kane	Ensemble de données en entrée.
HiSpOD	Choisi par l'utilisateur	Non	Oui	Oui	Oui	Non	Oui	Blast + critères de Kane	Base de données complète de CDS.
Metabolic Design	Choisi par l'utilisateur	Non	Non	Non	Non	Non	Oui	Blast + critères de Kane	Base de données complète de CDS.
CommOligo 2.0	Choisi par l'utilisateur	Oui	Oui	Oui	Oui	Non	Non	Alignement global + calcul thermodynamique + critères de kane	Ensemble de données en entrée.
OligoWiz 2.0	Choisi par l'utilisateur	Oui	Non	Non	Oui	Non	Non	Blast + thermodynamique + critères de kane	Un seul génome d'organisme.
ROSO	Choisi par l'utilisateur	Oui	Oui	Oui	Oui	Oui	Non	Blast	Fichier Fasta externe (typiquement un seul génome d'organisme).
ArrayOligoSelector	Choisi par l'utilisateur	Oui	Oui	Oui	Non	Non	Non	Blast + thermodynamique	Fichier Fasta externe (typiquement un seul génome d'organisme).
OligoArray 2.1	Choisi par l'utilisateur (15-75)	Oui	Oui	Oui	Oui	Non	Non	Blast + thermodynamique	Fichier Fasta externe (typiquement un seul génome d'organisme).
OligoPicker	Choisi par	Oui	Oui	Non	Oui	Non	Non	Blast	Ensemble de données en

	l'utilisateur (20-100)								entrée ou Fichier Fasta externe.
PROBEmer	Choisi par l'utilisateur	Oui	Non	Oui	Oui	Oui	Non	Suffix Tree	RCP (v8.1), Fichier Fasta externe
YODA	Choisi par l'utilisateur	Oui	Oui	Oui	Oui	Non	Non	Blast + thermodynamique	Fichier Fasta externe (typiquement un seul génome d'organisme).
ProbeSelect	Choisi par l'utilisateur	Oui	Oui	Non	Non	Oui	Non	Suffix Tree + thermodynamique	Un seul génome d'organisme
KASpOD	Choisi par l'utilisateur	Non	Non	Non	Non	Non	Oui	PATMAN	Base de données 16S extraite de Greengenes (pour les biopuces POA).

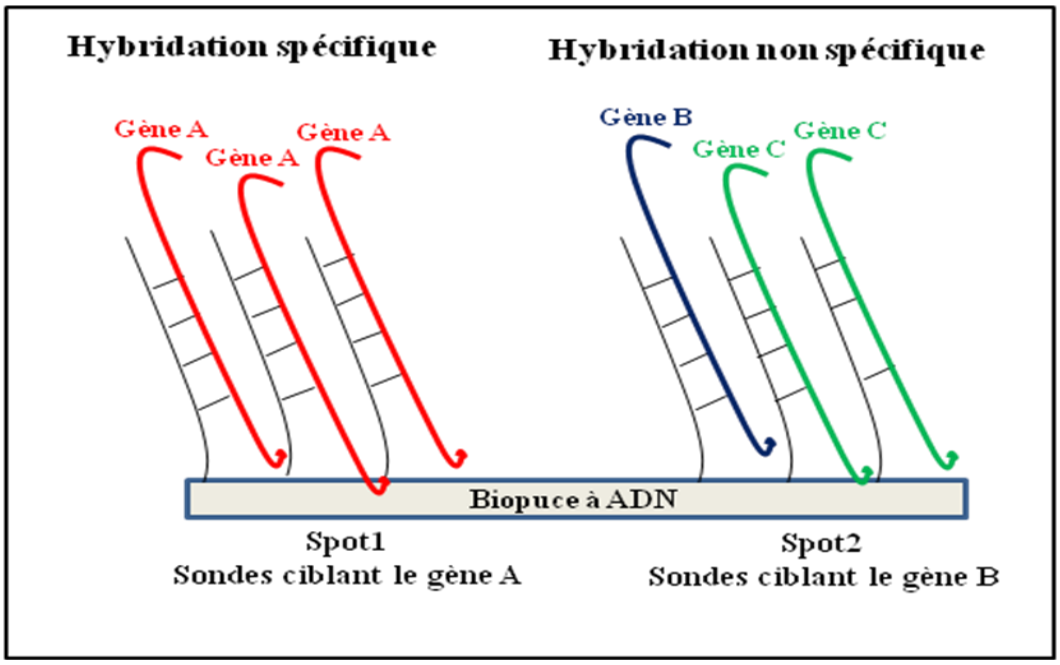

Figure 6. Différence entre hybridation spécifique et non-spécifique.

La complexité du problème pour assurer la détermination de sondes spécifiques est accentuée par l'impossibilité de prédire complètement le comportement de l'hybridation d'une sonde fixée sur un support solide comme dans le cas des biopuces ADN (Pozhitkov et al., 2007). Une sonde pourra s'hybrider avec une séquence non-cible si cette dernière contient une région très similaire à une portion de la sonde. Ainsi, il a été démontré que pour une sonde de 25-mers, les hybridations croisées peuvent être causées par la présence de fragments complémentaires à la sonde d'une longueur minimum de 10 à 16-mers (minimum 40% de la longueur de la sonde) (Wu et al., 2005). Une autre étude menée par Kane et al., (2000), estime qu'une sonde spécifique de 50-mers ne doit pas avoir une similarité de plus de 75 à 80% avec une séquence non ciblée sous peine de pouvoir engendrer des hybridations croisées. Ces recommandations de Kane pour l'évaluation des hybridations croisées sont largement utilisées dans les méthodes de sélection de sondes. Dans la même étude, il a été prouvé que pour une sonde de 50-mers, toute séquence présentant 15, 20 ou 35 nucléotides identiques avec cette sonde entraîne respectivement une intensité d'hybridation de 1%, 4% et 50% par rapport à l'hybridation complète de la séquence ciblée. De leur côté, Kucho et al. (2004), estiment que deux conditions doivent être validées pour avoir des sondes spécifiques de 45-mer: l'absence d'une identité sonde/séquence non cible supérieure à 71% et un pourcentage Guanine + cytosine (G+C) ne dépassant pas 55%. En effet, d'autres paramètres tels que le pourcentage des

nucléotides G et C d'une sonde, sa faible complexité, la présence d'homopolymères (suite d'un même nucléotide) ou encore les positions des mésappariements peuvent aussi affecter la spécificité des sondes (Wang and Seed, 2003; Huang et al., 2005; Leparc et al., 2009). Le dernier critère à prendre en compte pour la spécificité est la taille de la sonde. En effet, les oligonucléotides courts (18 - 25-mers) sont plus spécifiques que les oligonucléotides longs (50-mers ou plus). Cependant, les oligonucléotides longs seront plus sensibles et permettront une détection des séquences cibles faiblement représentées dans l'échantillon étudié.

La détection des hybridations croisées potentielles nécessite d'identifier les similarités entre la sonde et les séquences pouvant être présentent dans l'échantillon. Plusieurs algorithmes de recherche de similarité de séquences ont été proposés: FASTA (Pearson and Lipman, 1988), Blast (Altschul et al., 1990), MAFFT (Katoh et al., 2002), Kalign (Lassmann and Sonnhammer, 2005), PatMaN (Prüfer et al., 2008), etc. Ces algorithmes utilisent les alignements de séquences afin de calculer le score de similarité entre les séquences.

L'alignement de deux séquences est réalisé par la transposition d'une séquence contre l'autre en vérifiant les correspondances entre les caractères similaires. Pour calculer le score d'un alignement donné, trois cas de figures sont considérés: les appariements (« match »), les mésappariement (« mismatch »), et les gaps (figure 7). Un appariement entre deux séquences est une égalité entre les caractères des deux séquences à une même position de l'alignement, une valeur positive est alors associée. Un mésappariement est observé lorsque deux caractères (un de chaque séquence) à la même position sont différents, une valeur négative (<= 0) est alors associée. Si une séquence est alignée avec un espace inséré, un gap noté « - » se produit et une autre valeur négative lui est attribuée.

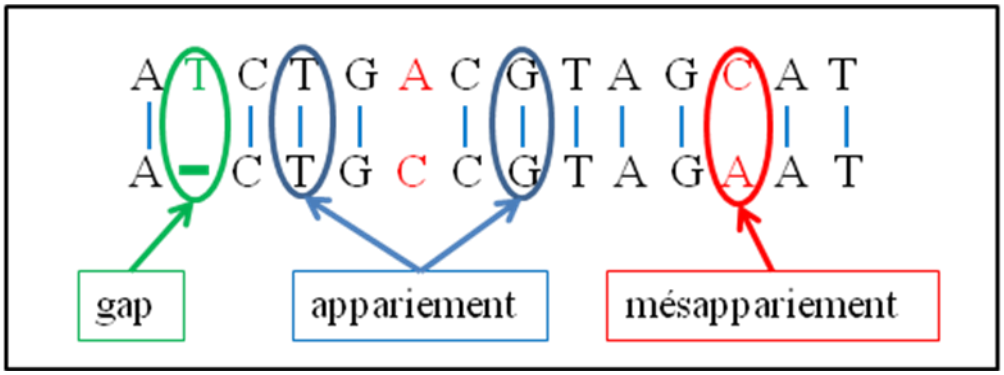

Figure 7. Exemple d'alignement entre deux séquences.

Actuellement, Blast (« Basic Local Alignment Search Tool ») (Altschul et al., 1990) est l'outil le plus utilisé pour la recherche de similarité par alignement local de séquences, du fait de sa rapidité et de la simplicité de son utilisation pour les biologistes. Plusieurs programmes distincts de recherche de similarité Blast existent, tels que:

- ✓ Blastn: compare une séquence nucléotidique contre une base de données de séquences nucléotidiques,
- ✓ Blastp: compare une séquence protéique contre une base de données de séquences protéique,
- ✓ Blastx: compare une séquence nucléotidique traduite dans les 6 phases de lecture contre une base de données de séquences protéique,
- ✓ tBlastn: compare une séquence protéique contre une base de données de séquences nucléotidiques traduites dans les 6 phases de lecture,
- ✓ tBlastx: compare une séquence nucléotidique traduite dans les 6 phases de lecture contre une base de données de séquences nucléotidiques traduites dans les 6 phases de lecture.

Blast utilise une méthode heuristique, assurant sa rapidité d'exécution au détriment d'une sensibilité parfois insuffisante. L'algorithme d'alignement dans Blast comporte 3 phases principales:

- ✓ **La recherche des mots voisins**: Blast suppose que pour obtenir un alignement significatif, les séquences alignées doivent partager des mots exacts (une série de lettres de longueur *W* fixée au départ). Toutes les positions de ces mots en commun entre deux séquences sont déterminées et seront utilisées comme base par la suite pour l'alignement. S'il n'y a pas de mots en commun dans l'alignement, le concept de mots voisins est alors introduit. Les voisins d'un mot incluent le mot lui-même et tous les autres mots ayant un score supérieur ou égal au seuil de score de voisinage *T* lorsqu'ils sont comparés au mot initial en utilisant une matrice de substitution. Le choix des paramètres *W* et *T* de la matrice de substitution est important pour contrôler la performance et la sensibilité du Blast.

- ✓ **L'extension des mots:** les mots obtenus précédemment sont étendus en parcourant les caractères proches des mots dans les deux sens. Cette extension va permettre d'améliorer ou de diminuer le score d'alignement en fonction de la similarité entre les deux séquences. Si ce score descend en dessous d'une valeur x donné par rapport à la valeur maximale qu'il avait atteint, si le score cumulé devient inférieur ou égal à zéro ou si la fin d'une des deux séquences est atteinte, le seuil l'extension est stoppée. Les alignements locaux ainsi trouvés sont appelés HSP (« High-scoring Segment Pair »).

- ✓ **L'évaluation:** les scores de similarité générés dans la deuxième phase sont ensuite évalués pour ne pas prendre en considérations les alignements non significatifs c'est-à-dire ceux pour lesquels les scores sont inférieurs au seuil de spécificité fixé par l'utilisateur.

Il est nécessaire de régler précisément les paramètres du Blast pour une utilisation optimale. Par exemple, il faut augmenter la valeur seuil de la *e-value* (« Expect value »: paramètre Blast qui décrit le nombre de résultats qu'on s'attendrait à trouver uniquement par hasard dans la base de donnée) pour assurer l'identification de similarité entre des séquences de petites tailles (sondes-cibles). En effet, la valeur de e value étant inversement proportionnelle au score d'alignement et le score étant dépendant de la longueur d'alignement, plus les séquences sont courtes plus le score sera faible. Il est aussi indispensable de réduire la taille *W* du mot utilisé dans la première phase de l'algorithme. En effet, les sondes oligonucléotidiques sont des séquences requêtes courtes pouvant présenter des mésappariements avec les séquences non ciblées, le paramètre *W* doit donc être fixé à la valeur minimale, soit 7 bases pour pouvoir identifier des similarités entre la sonde et ces séquences.

2.1.2. La sensibilité

Comme vu précédemment, la sensibilité augmente généralement avec la longueur des sondes. En effet, l'énergie de liaison sonde/cible pour les longues sondes est généralement plus élevée et les cinétiques d'hybridation sont irréversibles (Hughes et al., 2001; Relogio et al., 2002; Letowski et al., 2004). Par exemple, des sondes de 60-mers peuvent détecter leurs cibles avec sensibilité huit fois plus élevée que celles faisant 25-mers (Chou et al., 2004). Généralement, les POAs (« Phylogenetic Oligonucleotide

Arrays ») employées pour l'analyse des communautés microbiennes contiennent des sondes courtes (typiquement 24 - 25-mers) (Brodie et al., 2006; Paliy et al., 2009; Rajilic-Stojanovic et al., 2009), alors que les FGAs (« Functional Gene Arrays ») sont construites soit avec des oligunucléotides courts (15 à 30-mers) (Bodrossy et al., 2003; Stralis-Pavese et al., 2004) ou des oligonucléotides longs (40 à 70-mers) (Kane et al., 2000; Relogio et al., 2002; He et al., 2007). Pour pouvoir utiliser des sondes oligonucléotidiques courtes il est nécessaire dans la plupart des cas, d'amplifier les cibles par PCR. Cette étape assure un enrichissement des gènes ciblés et augmente donc la sensibilité, mais introduit de nombreux biais (Suzuki and Giovannoni, 1996; Peplies et al., 2004; Vora et al., 2004).

La formation de structures secondaires stables par les sondes et/ou les cibles est un autre facteur crucial qui doit être considéré pour minimiser la perte de sensibilité. Cependant, malgré une bonne connaissance des propriétés thermodynamiques de formation et de dissociation en solution des duplex d'acides nucléiques (SantaLucia et al., 1996), ainsi que la disponibilité de plusieurs algorithmes comme Mfold (Zuker, 2003) ou Hyther (Bommarito et al., 2000) pour leur prédiction précise, la formation de ces structures est difficilement quantifiable dans le contexte de biopuces. Cela est dû à la connaissance limitée sur la thermodynamique de l'hybridation sur des interfaces solide-liquide (Pozhitkov et al., 2006; Pozhitkov et al., 2007). Ainsi, plusieurs paramètres vont influencer la sensibilité dépendant de la longueur de la sonde et de sa composition.

2.1.2.1. La prédiction de la température de fusion et la thermodynamique des duplexes d'acides nucléiques

La température de fusion Tm (« melting temperature ») des acides nucléiques est la température à laquelle 50% des brins d'ADN sont dissociés et l'autre moitié est sous forme appariée. C'est donc la température à laquelle la moitié de l'ADN est sous forme monobrin et l'autre moitié sous forme double brin. En effet, deux molécules simple brin s'hybrident en une molécule double brin, tandis qu'une molécule double brin se dénature en deux molécules simple brin. Le passage d'une forme à l'autre est brutal en raison du caractère coopératif de la réaction.

La température de fusion dépend de nombreux facteurs tels que la longueur du fragment d'ADN, sa richesse en bases nucléotidiques C et G, ainsi que la concentration en ion Na+ du milieu réactionnel. Pour calculer la valeur de la Tm, plusieurs méthodes ont été proposées:

- ✓ **Méthode basique:** Elle considère uniquement le nombre d'occurrence de chaque nucléotide dans la séquence traitée (Marmur and Doty, 1962). En raison de sa simplicité, cette méthode est largement utilisée par les scientifiques malgré ses approximations.

  Pour des séquences d'une longueur maximale de 14-mers, la formule suivante est utilisée pour calculer la Tm:

  $$T_m = 2\,(wA + xT) + 4\,(yG + zC)$$

  Où *wA*, *xT*, *yG* et zC sont respectivement les nombres d'occurrences des bases A, T, G et C.

  Cependant, pour les séquences de longueur supérieure à 14-mers, la température de fusion est calculée par la formule suivante:

  $$T_m = 64.9 + 41 \times \left(\frac{yG + zC - 16.4}{wA + xT + yG + zC}\right)$$

  Ces deux équations supposent que la concentration en sel est égale à 50 mM, la concentration de l'oligonucléotide est égale à 50 nM et le pH est égal à 7.

- ✓ **Méthode d'ajustement:** Elle prend en considération la valeur de la concentration en sel dans le calcul de la température de fusion. Plusieurs formules ont été proposées en fonction de la longueur de la séquence. Selon Howley et al., (1979), pour une concentration en sel entre 0,01 et 0,4 M et un pourcentage des bases G+C entre 30% et 75%, la Tm est calculée suivant la formule suivante:

  $$T_m = 79.8 + 18.5 * log10([Na^+]) + \left(58.4 * \frac{yG + zC}{wA + xT + yG + zC}\right) + \left(11.8 * \left(\frac{yG + zC}{wA + xT + yG + zC}\right)^2\right) - \left(\frac{820}{wA + xT + yG + zC}\right)$$

  Cette équation suppose que la concentration de l'oligonucléotide est égale à 50 nM et le pH est égal à 7.

- ✓ **Méthode thermodynamique du plus proche voisin:** Elle est considérée comme la méthode la plus fiable. Elle prend en compte la composition en base, mais s'appuie également sur les paramètres thermodynamiques attribués à chaque paire de bases dépendant des bases voisines (Wetmur, 1991). Elle prend en compte la relation entre l'entropie ΔS, l'enthalpie ΔH, la concentration en sel et la concentration de

l'oligonucléotide. L'enthalpie *ΔH* représente la chaleur absorbée par la réaction lors de la formation ou de la dénaturation d'une paire de base à une pression constante. L'entropie *ΔS* est la mesure du degré de perte de liberté du système. L'énergie libre *ΔG* mesure la stabilité de la réaction chimique à une température Tm et à une pression constante. La valeur de *ΔG* est donnée par l'équation suivante:

$$\Delta G = \Delta H - T.\Delta S$$

La formule utilisée par la méthode du plus proche voisin pour le calcul du Tm est la suivante:

$$T_m = \frac{\Delta H - 3.4\ \frac{kcal}{^{\circ}K\ mole}}{\Delta S + R.\ln\left(\frac{1}{C}\right)} + 16.6\ log_{10}([Na^+]) - 273.15$$

Avec *R* la constante molaire des gaz parfaits (égale à 1,987 cal mol-1) et *C* la concentration molaire de la cible.

Cette formule suppose que l'hybridation est réalisée à un pH = 7, la concentration en sel est comprise entre 0,01 et 1,0 M, la séquence n'est pas symétrique et contient au moins un C ou un G, et que la taille de l'oligonucléotide est comprise entre 14 et 20 nucléotides afin d'obtenir des températures de fusion raisonnables.

Enfin, d'après (Panjkovich and Melo, 2005), des différences significatives sont observées pour les valeurs de Tm des oligonucléotides courts calculées en utilisant les différentes méthodes disponibles. Des données expérimentales supplémentaires couvrant un plus grand nombre d'oligonucléotides sont nécessaires pour évaluer l'exactitude des méthodes actuelles ou pour obtenir une estimation plus précise de la valeur de Tm expérimentale pour un oligonucléotide court. Par conséquent, l'usage d'un calcul consensus de la valeur de Tm avec uneprobabilité d'erreur minimale doit être proposé. Il devrait être dérivé de la comparaison des méthodes existantes sur un large ensemble de séquences. Les lignes directrices à suivre en vue d'accroître le succès des applications de la biologie moléculaire sont les suivantes:

(1) Les méthodes actuelles s'appliquent après avoir examiné les restrictions ou les limitations qu'elles ont (c'est-à-dire éviter les séquences qui forment une structure secondaire stable).

(2) Utiliser, si possible, des oligonucléotides possédant une composition moyenne en C+G et une longueur de 20 à 22 nucléotides (où la plupart des méthodes de calcul de Tm sont applicables)

(3) Utiliser une méthode de calcul de Tm consensus.

(4) Se référer à la littérature à venir pour des nouvelles méthodes améliorées pour la prédiction des valeurs de Tm.

2.1.2.2. Les structures secondaires

Tout comme un ADN (ou ARN) simple brin peut s'hybrider avec un brin complémentaire, il peut également se replier sur lui-même et s'hybrider avec ses propres bases lorsqu'il existe des zones complémentaires au sein de la même molécule (Zuker, 2000). On parle dans ce cas de structure secondaire. Une sonde peut donc s'hybrider avec elle-même sous des conditions thermodynamiques bien précises. Elle devient ainsi non accessible pour les cibles ce qui empêche ou affaiblie l'hybridation sonde/cible. Il en va de même pour les cibles formant alors des régions inaccessibles pour les sondes. La présence de structures secondaires peut donc être source d'erreurs au niveau des résultats d'hybridations. Il est important alors d'écarter, lors de la sélection des sondes, les oligonucléotides qui peuvent former des structures secondaires. La recherche de structures secondaires peut être considérée comme une étape supplémentaire pouvant intervenir à la fin du processus de détermination de sondes pour ne sélectionner que les plus efficaces.

Les structures secondaires peuvent exister sous plusieurs formes (figure 8). Ces structures sont d'autant plus stables que la valeur de leur énergie libre de formation est négative. Certaines méthodes expérimentales pour l'identification des structures secondaires ont été proposées telles que celles basées sur la cristallographie ou la RMN (résonance magnétique nucléaire). Cependant, ces méthodes demandent de grosses ressources en temps et en coût et ne sont pas applicables pour traiter des jeux de données toujours plus importants.

Des méthodes automatiques de prédiction des structures secondaires ont alors vu le jour. Ces méthodes reposent sur deux approches principales: i) l'approche comparative qui s'applique sur un ensemble homologue ou un alignement de séquences et qui est basée sur le fait que la structure secondaire est mieux conservée que la séquence (Eddy and Durbin, 1994; Gorodkin et al., 1997; Mathews and Turner, 2002), ii) l'approche thermodynamique qui se base sur des paramètres thermodynamiques pour calculer toutes les structures

possibles d'un ARN et rechercher ensuite la structure dont l'énergie est minimale (Stüber, 1986; Knudsen and Hein, 1999; Lyngso et al., 1999; Zuker, 2003; Knudsen and Hein, 2003; Ding et al., 2005; Do et al., 2006; Mathews, 2006; Wiese and Hendriks, 2006; Wiese et al., 2008). En effet à chaque structure correspond une quantité d'énergie libre, et la structure la plus stable est celle qui minimise l'énergie libre. Les premiers travaux basés sur cette approche sont ceux présentés dans (Tinoco et al., 1971). Les auteurs de cette étude ont établi des règles sur l'énergie des structures secondaires d'ARN et ont proposé une procédure pour prédire ces structures. Quelques années plus tard, Ninio (1979) a élargi les règles thermodynamiques par l'introduction des énergies des paires de bases non canoniques. Il a ensuite ajouté une dépendance de ces règles au contexte de l'appariement (Papanicolaou et al., 1984).

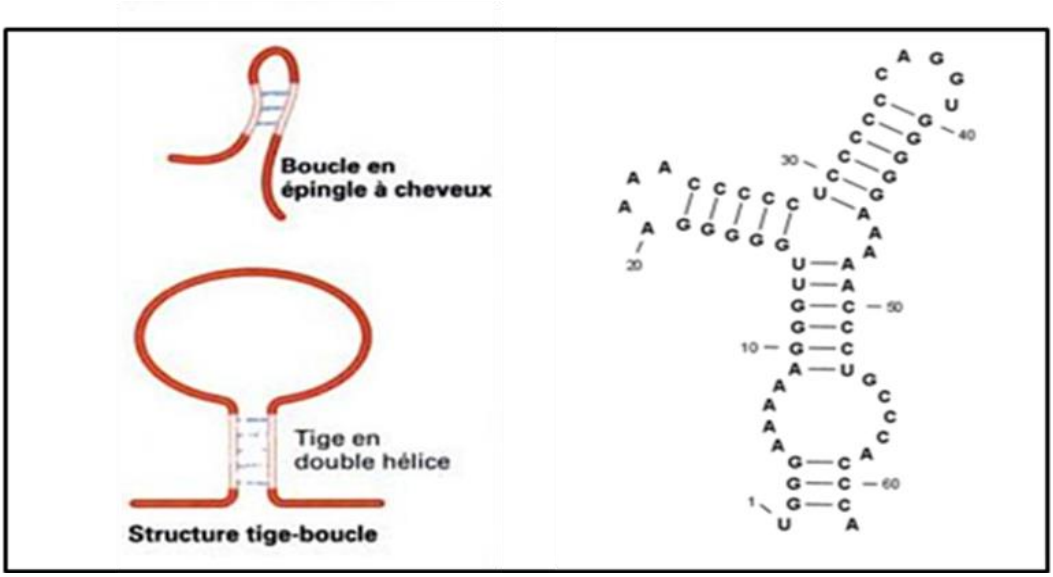

Figure 8: Exemples de structures secondaires des acides nucléiques (Missaoui, 2009).

La méthode de prédiction basée sur l'approche thermodynamique la plus utilisée actuellement est celle qui utilise la programmation dynamique. Le premier algorithme fondé sur la programmation dynamique a été proposé par Nussinov (Nussinov and Jacobson, 1980). Cet algorithme consiste à replier une séquence d'ARN en maximisant le nombre d'appariements. Il calcule la structure préférée pour des sous-séquences courtes. L'algorithme de Nussinov ne considère que les énergies des appariements. Dans cet algorithme, l'énergie d'une structure est obtenue comme étant la somme des énergies de chaque paire de bases qui la compose. Cependant, en réalité, l'énergie libre ne dépend pas seulement des appariements des paires de bases mais aussi des empilements, des boucles

internes, terminales et multiples, des épingles à cheveux et des renflements. Une amélioration de cet algorithme a été alors proposée par (Zuker and Stiegler, 1981). En effet, les auteurs ont pris en considération les liaisons hydrogènes, les énergies d'empilements et les énergies structurales plus globales. Zuker (1989) a ensuite amélioré cet algorithme en optant pour une solution déterministe qui calcule toutes les solutions alternatives en parallèle. Cet algorithme a été implémenté dans le logiciel Mfold (Zuker, 2003) qui est actuellement le logiciel le plus utilisé pour la prédiction des structures secondaires d'une séquence ARN.

La plupart des algorithmes basés sur la programmation dynamique ne permettent pas la recherche des pseudonœuds. Un algorithme permettant la prédiction des structures secondaires incluant des pseudonœuds a été proposé dans (Rivas and Eddy, 1999). Cependant, cet algorithme a une complexité très élevée. D'autres algorithmes utilisant l'approche thermodynamique sans programmation dynamique mais permettant de rechercher des pseudonœuds avec des complexités raisonnables ont été donc proposés. C'est le cas des trois algorithmes intégrés dans la plateforme STAR (« STructure Analysis of RNA »): le premier est basé sur un algorithme glouton (Abrahams et al., 1990), le deuxième (Gultyaev, 1991) est basé sur un algorithme stochastique et une procédure de Monte Carlo, et le dernier (Van Batenburg et al., 1995; Gultyaev et al., 1995) se base sur un algorithme génétique.

Les règles thermodynamiques les plus utilisées actuellement sont celles établies par (Xia et al., 1998; Mathews et al., 1999) et révisées par (Mathews et al., 2004). Ces règles ont été implémentées dans des programmes de prédiction de structures secondaires tels que RNAfold (Hofacker, 2003) qui repose sur l'algorithme de Nussinov (Nussinov and Jacobson, 1980) et sur des fonctions de partition (McCaskill, 1990) permettant d'attribuer des probabilités à tous les appariements possibles.

Malgré tous les efforts d'optimisation, les méthodes de recherche de structures secondaires restent toujours des prédictions qui sont seulement capables de prédire correctement que 50 à 70 % structures secondaires. Elles restent donc insuffisantes pour obtenir des résultats optimaux.

2.1.3. L'uniformité

Parce que le principe de la technologie des biopuces repose sur l'hybridation simultanée de plusieurs sondes dans les mêmes conditions (concentration en sel, température, etc.), il est important de s'assurer que les sondes sélectionnées ont des

comportements thermodynamiques aussi uniformes que possible (Loy and Bodrossy, 2006; Wagner et al., 2007). La meilleure façon d'y parvenir est de sélectionner des sondes ayant des propriétés structurelles homogènes telles que la longueur de la sonde, le contenu G+C, la température de fusion (Tm) ou les capacités de liaison (ΔG).

2.2. Développement des biopuces phylogénétiques à oligonucléotides

Afin d'étudier rapidement les communautés procaryotes présents dans des environnements complexes, les biopuces phylogénétiques oligonucléotidiques POAs (« Phylogenetic Oligonucleotide Arrays ») utilisent le biomarqueur de la petite sous unité de l'ARNr (Wilson et al., 2002; Brodie et al., 2006; Palmer et al., 2006; DeSantis et al., 2007).

Le principal obstacle à la conception d'un POA est celui des hybridations croisées potentielles. En effet, dans de nombreux cas, les séquences des gènes codant l'ARNr 16S sont très conservées et rendent difficile la conception de sondes discriminatoires au niveau de l'espèce (Bae and Park, 2006). Pour contourner ce problème, une conception hiérarchique permet de sélectionner des sondes pour des taxons microbiens à des niveaux phylogénétiques supérieurs (Huyghe et al., 2008; Liles et al., 2010).

2.2.1. Sélection de sondes pour les biopuces POAs

Des approches manuelles ainsi que des logiciels automatiques ont été développés pour concevoir des POAs (« Phylogenetic Oligonucleotide Arrays ») tout en prenant en compte les principaux critères de sélection de sondes efficaces, à savoir la sensibilité et la spécificité. Actuellement, trois programmes sont disponibles et capables de fonctionner avec des données structurées pour extraire et analyser des séquences de bases de données dédiées et proposer une sélection de sondes phylogénétiques ciblant le gène exprimant l'ARNr 16S.

Le premier programme est l'outil de sélection de sondes inclus dans la suite logicielle ARB (Ludwig et al., 2004) qui est couramment utilisé pour sélectionner des oligonucléotides de longueur comprise en 10 et 100-mers. La première étape du programme consiste à sélectionner un groupe cible. Dans un second temps, l'algorithme identifie les tronçons de séquences uniques qui pourraient servir de sites cibles, et retourne ensuite une liste triée des oligonucléotides potentiels. Enfin, les sondes proposées peuvent être comparées à toutes les séquences de la base de données en utilisant le logiciel « Probe

Match » d'ARB pour identifier des hybridations croisées potentielles. ARB propose également différents ensembles de sondes prédéfinis, chacun ciblant des groupes phylogénétiques distincts. Cet outil a été largement utilisé pour développer des POAs à faible densité c'est-à-dire ne contenant que quelques centaines de sondes oligonucléotides. Ces sondes ciblent généralement soit des groupes restreints de microorganismes impliqués dans un métabolisme spécifique (Loy et al., 2002; Kelly et al., 2005; Franke-Whittle et al., 2009), ou appartenant à un taxon spécifique (Castiglioni et al., 2004; Lehner et al., 2005; Loy et al., 2005; Kyselkova et al., 2008; Schonmann et al., 2009; Liles et al., 2010), ou vivant dans un habitat/écosystème d'un intérêt particulier (Neufeld et al., 2006; Sanguin et al., 2009).

Le deuxième programme est PRIMROSE (Ashelford et al., 2002), une application qui utilise des bases de données standards ou sur mesure, et permet la détermination de sondes dégénérées. Initialement, un alignement multiple est produit en utilisant les différentes séquences représentant un taxon donné. Chaque sonde est ensuite testée contre toutes les séquences de la base de données initiale, pour caractériser les hybridations croisées potentielles, et pour vérifier une bonne couverture du taxon ciblé. Bien que cet outil ait été développé pour identifier des sondes phylogénétiques et des amorces, il a été principalement utilisé dans des approches FISH (Fluorescent In Situ Hybridization) et d'autres utilisant la PCR (Rusch and Amend, 2004; Yu et al., 2005; Feldhaar et al. 2007; Boeckaert et al., 2008; Klitgaard et al., 2008; Mühling et al., 2008; Gittel et al., 2009; Fraune et al., 2010; Bers et al., 2011). Peu d'applications POAs utilisant PRIMROSE ont été rapportées. Blaskovic et Barak (2005) ont présenté le développement d'une biopuce de faible complexité pour détecter spécifiquement des bactéries transmises par les tiques responsables de certaines maladies humaines et animales.

Le troisième programme est ORMA (« Oligonucleotide Retrieving for Molecular Applications ») qui représente une bonne solution pour concevoir des sondes très discriminantes (Severgnini et al., 2009). Bien qu'il ait été d'abord appliqué à la sélection de sondes ciblant les gènes exprimant les ARNr 16S, ce logiciel peut être utilisé sur n'importe quel jeu de séquences. En utilisant ce programme, Candela et al. (2010) ont conçu la biopuce « HTF-Microbi.Array » qui permet l'analyse génétique de la flore microbienne intestinale humaine à des niveaux taxonomiques différents. Elle est composée de 30 paires de sondes ciblant 30 groupes appartenant à différents niveaux phylogénétiques: espèces, genre, famille, cluster ou groupes d'espèces.

En outre, d'autres approches de détermination de sondes, qui ne sont pas entièrement implémentées et automatisées dans des programmes, ont été proposées pour le développement de POAs de haute densité permettant de contrôler tous les taxons connus des bactéries ou des archées à l'aide d'une seule biopuce (Wilson et al., 2002; DeSantis et al., 2007). Ces approches reposent sur des algorithmes sophistiqués pour la sélection d'une multitude de sondes et l'analyse de profils d'hybridation très complexes. Le meilleur exemple est la biopuce « PhyloChip » développée par Brodie et al. (2006) en utilisant la stratégie de design de sondes définie par (Desantis et al., 2003) et basée sur la plate-forme de biopuces « Affymetrix GeneChip ». Cette biopuce est capable d'identifier simultanément des milliers de taxons présents dans un échantillon environnemental (voir chapitre 1 section 4.2). Notons, que dans l'approche Affymetrix une sonde s'hybridant parfaitement à la cible est sélectionnée ainsi qu'une sonde montrant un mésappariement central avec la cible dans le but de discriminer les hybridations imparfaites.

La principale limitation de toutes les stratégies proposées pour la détermination de sondes est qu'elles ne garantissent que l'étude des microorganismes pour lesquels des séquences sont disponibles dans les bases de données publiques. Malheureusement, pour une grande majorité des espèces microbiennes aucune séquence n'est disponible. Un défi majeur pour l'avenir est l'amélioration de la technologie des biopuces à ADN pour s'appuyer sur de nouvelles stratégies de détermination de sondes exploratoires ciblant de nouvelles espèces qui ne sont pas encore répertoriées dans les bases de données.

2.2.2. Stratégies de détermination de sondes exploratoires

Le concept de sondes multiples consiste à utiliser plusieurs sondes ciblant un organisme à des niveaux phylogénétiques/taxonomiques différents. La détermination de sondes en utilisant ce concept réduit considérablement le risque d'erreur d'identification, et permet souvent la discrimination de bactéries jusqu'au niveau de l'espèce (Ludwig et al., 1998; Loy and Bodrossy, 2006; Schliep and Rahmann, 2006; Huyghe et al., 2008; Schonmann et al., 2009; Liles et al., 2010). Les biopuces construites en se basant sur ce concept peuvent détecter la présence des taxons inconnus sans pour autant pouvoir les identifier.

Actuellement, seuls deux logiciels dédiés aux POAs offrent la possibilité de sélectionner des sondes exploratoires discriminantes, à savoir KASpOD (Parisot et al., 2012) et PhylArray (Militon et al., 2007). KASpOD est un service web pour la

détermination d'oligonucléotides spécifiques et exploratoires. L'algorithme de sélection de sondes dans KASpOD est implémenté en langage Perl. La stratégie KASpOD consiste à extraire les séquences k-mers à partir d'un groupe de séquences ciblées ainsi qu'à partir d'un autre groupe contenant toutes les séquences non ciblées, en utilisant Jellyfish version 1.1.4 (Marcais and Kingsford, 2011). Chaque séquence k-mers trouvée en même temps dans les deux groupes est supprimée pour garder seulement les séquences k-mers non redondantes. Ces séquences sont ensuite classées à l'aide de CD-HIT version 4.5.4 (Li and Godzik, 2006) avec un seuil d'identité de 88%, et une séquence consensus pouvant être dégénérée est construite pour chaque cluster. La couverture (le pourcentage de séquences détectables par la sonde) de ces séquences est alors testée contre le groupe de séquences ciblées, et leur spécificité est quant à elle évaluée contre le groupe de séquences non ciblées. L'avantage de ce logiciel est qu'il est indépendant de l'alignement préalable des séquences pour la détermination des sondes. Ainsi, il peut traiter efficacement des masses de données beaucoup plus conséquentes. De son côté, le programme PhylArray (Militon et al., 2007), développé en Perl, permet de sélectionner des sondes sur la base d'alignements de séquences. La première étape de la détermination de sondes dans PhylArray est l'extraction de toutes les séquences disponibles correspondantes à un taxon donné à partir d'une base de données personnalisée. Dans un second temps, un alignement multiple de séquences est effectué en utilisant l'algorithme ClustalW (Thompson et al., 1994). Dans un troisième temps, une séquence consensus dégénérée est produite en tenant compte de la variabilité de séquence pour chaque position, ce qui permet la détermination de sondes dégénérées. Enfin, toutes les combinaisons possibles de chaque sonde dégénérée sont déduites et leur spécificité est ensuite vérifiée pour déterminer les hybridations croisées potentielles avec les séquences de la base de données initialement conçue. Parmi les combinaisons dérivées de chaque sonde dégénérée, certaines correspondent à des séquences qui n'étaient pas présentes dans les bases de données publiques (figure 9). Ces sondes doivent, par conséquent, permettre l'exploration de la fraction non encore décrite des communautés microbiennes environnementales correspond au groupe taxonomique ciblé. Les évaluations expérimentales comparatives r ont montré que les sondes conçues avec PhylArray donnent une plus grande sensibilité et spécificité que celles conçues avec les logiciels PRIMROSE et ARB précédemment décrits (Militon et al., 2007). Cependant, PhylArray peut nécessiter jusqu'à plusieurs jours de calcul pour sélectionner des sondes de 25-mers ciblant un seul organisme contenant plusieurs centaines de séquences 16S de

longueur 1200 – 1500 bases, même en utilisant sa version parallèle disponible sur internet et utilisant un cluster de calcul (Missaoui, 2009).

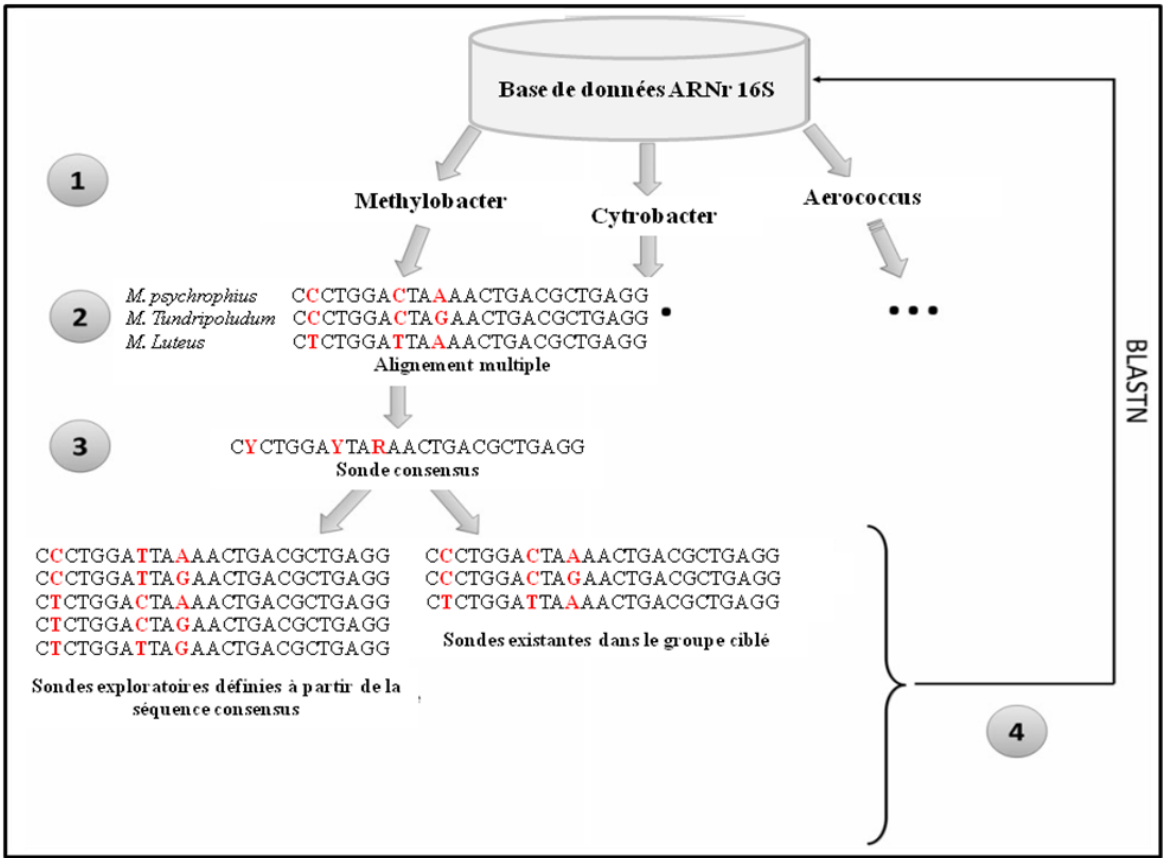

Figure 9. Etapes de détermination de sondes de l'algorithme PhylArray: 1: Sélection des séquences appartenant à un groupe cible, **2:** alignement multiple de ces séquences, **3:** détermination de séquences consensus spécifiques du groupe, **4:** test de spécificité des sondes déduites de la sonde dégénérée.

Pour une meilleure utilisation de ce logiciel, les auteurs de PhylArray ont construit une base de données de séquences ARNr 16S. Cette base spécialisée a été établie à partir de la banque de données de EMBL en prenant en compte des paramètres de qualité des séquences 16S (longueur, composition et affiliation) pour ne conserver que des séquences non redondantes et de bonne qualité et donc assurer une meilleure sélection de sondes.

Une biopuce développée avec la stratégie de PhylArray a été utilisée pour évaluer la diversité bactérienne dans deux sols différents (Delmont et al., 2011). Les auteurs ont mis en évidence l'influence significative de plusieurs paramètres comme la profondeur de l'échantillonnage et les protocoles d'extraction de l'ADN sur l'estimation de la

biodiversité. Une autre biopuce permettant d'explorer le microbiote intestinal humain au niveau de la famille a également été développée (Tottey et al., 2013).

2.3. Développement de biopuces à ADN fonctionnelles

La difficulté de conception des biopuces fonctionnelles FGAs (« Functional Gene Arrays ») est de pouvoir déterminer des sondes ciblant des familles de gènes pouvant être plus ou moins conservées. Là encore, du fait de l'extraordinaire diversité des microorganismes, seule une petite fraction des gènes est actuellement connue. La détermination de sondes exploratoires est une fois encore un enjeu majeur.

2.3.1. Détermination de sondes à partir d'un alignement de séquences nucléiques

De nombreux programmes de détermination de sondes sont actuellement librement accessibles (Lemoine et al., 2009). La plupart d'entre eux ont été développés pour étudier un seul génome, et par conséquent, se limitent à la détermination des sondes ciblant une seule séquence par gène. En revanche, peu de stratégies offrent la possibilité de concevoir des sondes permettant une large couverture de plusieurs variants de séquences pour une famille de gènes donnée.

Avec l'accroissement des données de séquences correspondant à des gènes fonctionnels (séquençage complet de génomes, études ciblant des marqueurs fonctionnels spécifiques et études métagénomiques), de nouveaux programmes prenant en compte cette grande diversité ont été développés. HPD (« Hierarchical Probe Design ») est le premier programme proposé dédié à la détermination des oligonucléotides pour biopuces fonctionnelles. Il a été développé avec le langage Pascal Objet sous Windows (version Delphi7 de Borland). HPD se base sur un alignement multiple et une classification hiérarchique des séquences (Chung et al., 2005). En effet, il réalise d'abord l'alignement (avec ClustalW (Thompson et al., 1994)) et le regroupement hiérarchique des séquences d'entrée en utilisant les méthodes de « neighbor-joining » (Saitou and Nei, 1987) ou d'UPGMA (Sokal and Michener, 1958) afin de générer l'ensemble complet des sondes candidates possibles. L'ensemble optimal de sondes est ensuite déterminé en fonction des critères de qualité de la sonde comme la couverture du cluster, la spécificité et la composition en G+C. Bien que cet outil ne soit pas exploratoire, il produit automatiquement des sondes contre tous les nœuds hiérarchiques, offrant une couverture étendue des variants connus à partir d'un gène fonctionnel conservé.

Le programme ProDesign, est un logiciel de sélection de sondes pour la détection de gènes ou de familles de gènes dans des échantillons environnementaux. Il a été développé par Feng et Tillier (2007) et est implémenté avec le langage de programmation C. ProDesign prend en entrée un ensemble de séquences et une liste initiale de groupes auxquelles ces séquences appartiennent. La sélection de sondes se fait en 4 étapes. Premièrement, les listes de mots sont construites en utilisant un algorithme de hachage de graines espacées. Le but ici est de trouver tous les mots possibles spécifiques à chaque groupe de séquences. Ensuite, en utilisant ces listes de mots, des sondes candidates pour chaque groupe sont déterminées. Les sondes d'un groupe se composent d'un ou de plusieurs mot spécifiques à ce groupe. Ces mots peuvent être chevauchants ou séparés par de petits gaps. Toutes les sondes candidates sont filtrées pour vérifier leur sensibilité et supprimer celles qui possèdent un pourcentage de G+C très élevé ou très bas, une valeur extrême de température de fusion ou des structures secondaires. La troisième étape consiste à regrouper les groupes de séquences pour lesquels aucune sonde n'a été sélectionnée, puis, sélectionner des sondes pour ces nouveaux groupes en utilisant la même méthode de l'étape 2. Cette étape peut être répétée en cas de besoin. Enfin, la dernière étape consiste à sélectionner le meilleur ensemble de sondes en se basant sur les valeurs de température de fusion et les propriétés d'hybridation déterminées en utilisant le package OligoArrayAux[1].

Comme avec HPD, cet outil ne fournit pas de sondes ciblant les séquences d'acides nucléiques non encore répertoriées. En outre, à notre connaissance, aucune application utilisant cette stratégie n'a été rapportée dans la littérature.

Ces stratégies permettent à une plus large gamme de variants de séquences d'être couverte, et semblent donc être mieux adaptées pour décrire les communautés microbiennes des environnements complexes. Cependant, leurs principaux inconvénients sont leur incapacité à générer des sondes exploratoires et l'absence de test de spécificité pour la recherche des hybridations croisées potentielles contre une base de données complète représentative de la diversité microbienne. Au sein de notre équipe, un algorithme de détermination de sondes pour biopuces fonctionnelles, appelé HiSpOD (« High Specific Oligo Design »), a été proposé pour résoudre ce problème (Dugat-Bony et al., 2011). HiSpOD a été implémenté en Perl. HiSpOD est le seul programme capable de

[1] http://mfold.rna.albany.edu/?q=DINAMelt/OligoArrayAux

prendre en compte à la fois les paramètres primordiaux nécessaires pour la sélection de sondes sensibles et spécifiques (taille, Tm, % G+C, complexité) et la détermination de sondes dégénérées pour la création de sondes exploratoires (figure 10). D'autre part, la spécificité de toutes les sondes sélectionnées est vérifiée contre une base de données complète dédiée aux communautés microbiennes, appelée EnvExBase (« Environmental Expressed sequences dataBase »). Pour construire EnvExBase, toutes les séquences d'ADN codantes (CDS) existantes dans les divisions PRO (procaryotes), FUN (champignons) et ENV (environnementales) de la banque de données EMBL ont été extraites et vérifiées pour enlever les séquences de mauvaise qualité. La base de données ainsi obtenue contient 9.129.323 séquences. A notre connaissance, EnvExBase est la première base de données CDS dédiée à l'écologie microbienne (Dugat-Bony et al., 2011).

Pour valider la stratégie HISpOD, une biopuce FGA composée de 295 sondes de 50-mers et ciblant 21 gènes ou familles de gènes impliqués dans biodégradation des chloroéthènes a été développée (Dugat-Bony et al., 2011). Les différents tests d'hybridation de cette biopuce ont démontré l'efficacité d'HiSpOD et la puissance des sondes déterminées en termes de spécificité et de sensibilité, tout en assurant une quantification des gènes ciblés.

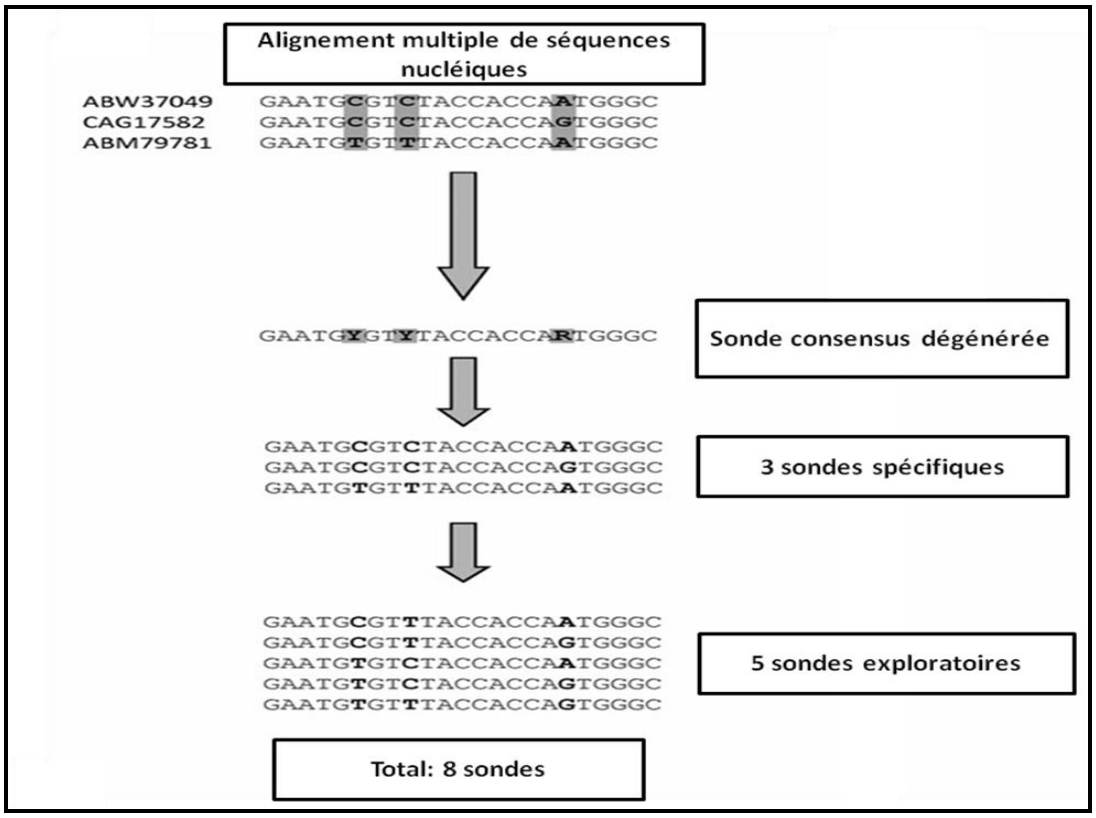

Figure 10. Stratégie de design des sondes exploratoires avec HiSpOD

2.3.2. Détermination de sondes à partir de séquences protéiques

Contrairement aux stratégies décrites ci-dessus, un certain nombre de nouvelles stratégies ont été proposées pour initier la détermination des sondes non plus à partir des séquences d'acides nucléiques, mais de régions peptidiques conservées. Toutes les séquences nucléiques pouvant coder pour ces peptides sont alors définies.

La première stratégie basée sur ce principe a été décrite par Bontemps et al. (2005) et a été appelée CODEHMOP (« COnsensus DEgenerate Hybrid Motif Oligonucleotide Probe »). Il s'agit d'une adaptation de la stratégie CODEHOP (« COnsensus DEgenerate Hybrid Oligonucleotide Primer ») pour le design des amorces PCR, développée à l'origine pour identifier des gènes phylogénétiquement éloignés codant pour des protéines qui appartiennent à des familles connues (Rose et al., 1998; Rose et al., 2003; Boyce et al., 2009). Dans la stratégie de CODEHMOP, des motifs d'acides aminés conservés sont identifiés à partir des alignements multiples de séquences protéiques. Ensuite, toutes les combinaisons nucléiques possibles (15 – 21-mers) de la région la plus conservée (5 - 7 acides aminés) de chaque motif de protéine, sont recréés et flanqués de 12 - 15 nucléotides à chaque extrémité (5' end et 3' end). Les sondes finales se composent donc d'un noyau

central variable pour cibler une diversité plus grande et deux séquences ajoutées aux extrémités de la sonde pour augmenter sa longueur. Les auteurs ont utilisé cette approche pour développer une biopuce à ADN prototype couvrant toutes les séquences *nodC* (gène de nodulation) décrites et non décrites chez les bactéries, et l'appliquer aux nodules de légumineuses (Bontemps et al., 2005). Cette stratégie a permis aux auteurs de détecter de nouvelles séquences *nodC* présentant moins de 74% d'identité avec les séquences connues.

L'utilisation de la stratégie CODEHMOP est limitée par le fait qu'elle n'est pas implémentée dans un programme entièrement automatique et qu'aucun test de spécificité des sondes n'est incorporé. Néanmoins, cette approche semble particulièrement adaptée pour cibler l'ensemble des variants pouvant coder pour une même protéine.

Terrat et al. (2010) ont développé un logiciel écrit en Java et Perl, appelé « Metabolic Design », qui assure la reconstruction *in silico* des voies métaboliques, l'identification des motifs conservés à partir des alignements multiples de protéines, et la génération de sondes exploratoires efficaces via une interface graphique simple et conviviale. Pour utiliser « Metabolic Design », tout d'abord et avant l'étape du design des sondes, l'utilisateur reconstruit *in silico* une voie métabolique d'intérêt avec les substrats et les produits provenant de chaque étape métabolique. Une enzyme de référence pour chacune de ces étapes est sélectionnée et sa séquence protéine extraite d'une base de données (par défaut, Swiss-Prot). Cette séquence est ensuite utilisée pour rechercher par similarité toutes les protéines appartenant à la même famille au sein des bases de données Swiss-Prot et TrEMBL. Après avoir sélectionné les séquences les plus représentatives, elles sont alignées pour débuter la phase de détermination des sondes. La traduction inverse des acides aminés est réalisée pour chaque site moléculaire identifié, en tenant compte de la redondance du code génétique, pour produire une séquence consensus nucléique dégénérée. Toutes les sondes dégénérées qui répondent aux critères définis par l'utilisateur (longueur de la sonde et taux de dégénérescence maximal) sont conservées. Toutes les séquences spécifiques déduites de chaque sonde dégénérée sont ensuite extraites et les hybridations croisées potentielles qu'elles peuvent engendrer sont vérifiées contre une base de données représentative (comme par exemple la base EnvExBase du programme HiSpOD). Enfin, le fichier de sortie, énumérant toutes les sondes dégénérées sélectionnées par l'utilisateur, permet de déduire toutes les combinaisons possibles et de les organiser en sondes spécifiques ou exploratoires (figure 11). L'approche a été validée par l'étude des

enzymes impliquées dans la dégradation des hydrocarbures aromatiques polycycliques (Terrat et al., 2010).

Une approche complémentaire pour la détermination de sondes oligonucléotidiques qui combine une excellente spécificité avec une sensibilité potentiellement élevée, est l'utilisation de la stratégie GoArrays développée par Rimour et al. (2005) (logiciel disponible sur http://g2im.u-clermont1.fr/serimour/goarrays.html). Dans cette approche, la sonde oligonucléotidique est une concaténation, *via* une séquence de liaison (linker) aléatoire et courte (de 3 à 6-mers), de deux sous-séquences courtes qui sont complémentaires à des régions disjointes de la cible (figure 12). Cette stratégie a été conçue pour améliorer l'efficacité des biopuces fonctionnelles et taxonomiques pour une large gamme d'applications (Rimour et al., 2005; Zhou et al., 2007; Pariset et al., 2009; Kang et al., 2010).

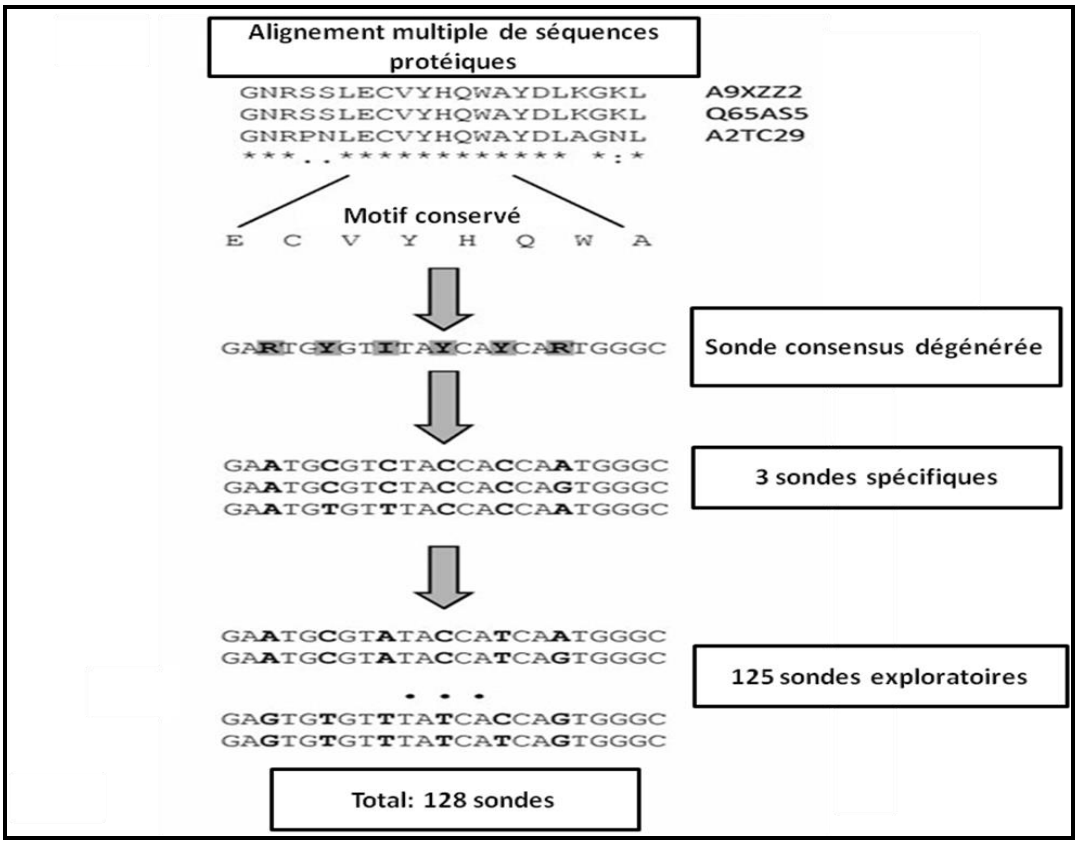

Figure 11. Stratégie de design des sondes exploratoires avec Metabolic Design.

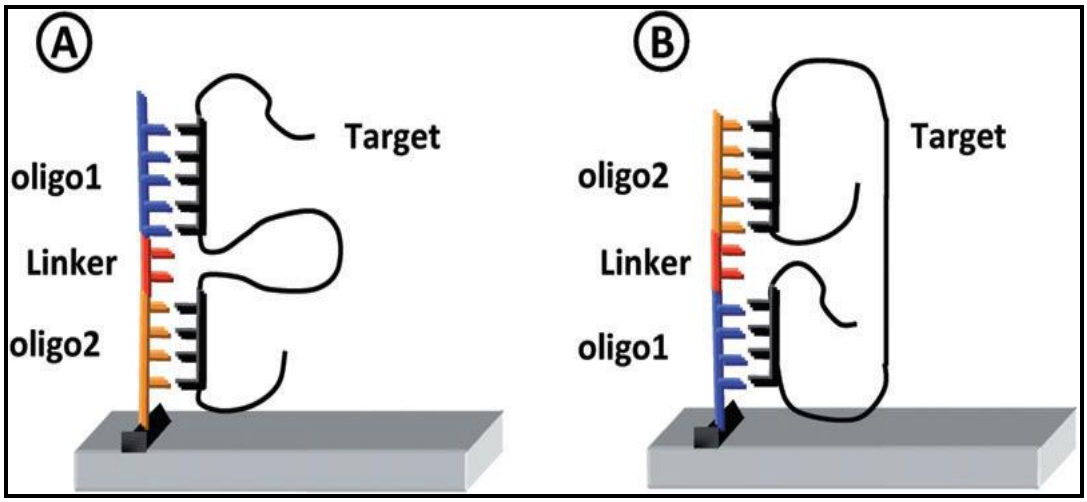

Figure 12. Principe de la stratégie GoArrays: Dans cette stratégie, deux sondes courtes spécifiques sont concaténées via un linker pour former une sonde oligonucléotide combinant la très bonne spécificité des sondes courtes et la sensibilité des sondes longues. En fonction des positions des sondes, la cible peut former deux types de boucles stables lors de l'hybridation (A et B).

Quel que soit l'outil de sélection de sondes utilisé, les biopuces à ADN en écologie microbienne sont complexes et nécessitent des logiciels d'analyse adaptés pouvant traiter la grande quantité de données issues de cette technique.

3. Analyse des données de biopuces ADN pour l'écologie microbienne

Ce processus consiste en plusieurs étapes successives: la lecture des signaux d'hybridation à l'aide d'un scanner, l'extraction des données numériques, la normalisation des données extraites, et enfin l'analyse de ces données. Les deux premières étapes sont généralement réalisées à l'aide des outils et matériels fournis par les fabricants des biopuces utilisées (voir chapitre 1 section 4.4). Cependant, pour la normalisation et l'analyse des données numériques, un grand nombre d'approches a été proposé. Ces techniques dépendent souvent du type de la biopuce et de son application.

3.1. Normalisation des données numériques issues des biopuces à ADN

Du fait de la complexité de mise en œuvre des différentes étapes d'une expérience d'hybridation de biopuces, de nombreux biais techniques peuvent être introduits tout au

long de l'expérience, de la préparation des cibles à l'analyse des images, en passant par l'hybridation (l'efficacité d'incorporation des fluorochromes utilisés, l'utilisation de plusieurs aiguilles pour déposer les sondes sur les lames, etc.) (Hartemink et al., 2001). Ces biais peuvent provoquer des variations systématiques affectant les intensités mesurées qui se confondent avec le signal biologique à étudier. La quantification et la soustraction de ces variations, par normalisation, permet de mieux interpréter les résultats des biopuces.

Dans ce contexte, plusieurs approches de normalisation des données de biopuces ont été développées (Bilban et al., 2002; Quackenbush, 2002; Schuchhardt et al., 2000; Yang et al., 2002; Park et al., 2003; Bolstad et al., 2003; Fujita et al., 2006; Dabney and Storey, 2007; Oshlack et al., 2007; Xiong et al., 2008; Baber et al., 2011; Kang and Xu, 2013) mais aucune méthode n'est universelle. Chacune possède des avantages et inconvénients et doit être considérée en fonction du type de biopuce ADN et des questionnements biologiques. Parmi les techniques de normalisation les plus utilisées, on trouve la normalisation globale, la normalisation par régression linéaire, les méthodes par « invariant de rang » (Tseng et al., 2001), les techniques basées sur les statistiques des ratios (Kerr et al., 2000; Wolfinger et al., 2001), la technique LOWESS (« Locally weighted scatter plot smoothing ») (Cleveland, 1979, 1981) et les méthodes d'ajustement par quantile (Bolstad et al., 2003). Il est possible de distinguer deux grands types d'approches de normalisation: linéaires et non linéaires. Chacune de ces méthodes peut être appliquée au niveau local ou global. Dans le mode linéaire, la normalisation consiste à trouver une constante de proportionnalité entre les données des fluorochromes Cy3 et Cy5. Les approches non linéaires sont quant à elles plus générales et considérées comme plus puissantes.

Les méthodes de normalisation globale sont les premières qui ont été développées. La principale méthode globale est celle de la normalisation par rapport à la moyenne (ou la médiane) globale des intensités. Cette méthode consiste à ajuster les intensités des sondes afin que la moyenne des logarithmes des ratios Cy3/Cy5 soit égale à 0 sur l'ensemble de la biopuce. Cette méthode suppose que les gènes étudiés constituent un échantillon aléatoire de l'ensemble des gènes de l'organisme, et que la majorité de ces gènes s'expriment de la même façon dans les deux conditions de l'expérience émettant ainsi un signal identique en Cy3 et Cy5.

Les méthodes LOWESS et d'ajustement par quantile sont les plus couramment utilisées pour la normalisation des données de biopuces (Saviozzi et al., 2006; Fujita et al., 2006; Oshlack et al., 2007; Dabney and Storey, 2007). La méthode Lowess a été proposée

afin de corriger le biais systématique qui existe dans les valeurs des logarithmes des ratios de fluorescence (Yang et al., 2002). Elle permet de calculer une courbe de normalisation ajustée à la forme du nuage par régression linéaire locale, grâce à un système de « fenêtre glissante ». En ce qui concerne la méthode de quantile, elle a l'avantage d'être plus rapide et plus simple que LOWESS car elle ne demande pas de réglage de paramètres. Le but de la méthode des quantiles est de rendre identique la distribution des intensités de sondes pour chaque réseau de sondes présent sur la biopuce ADN. Cette méthode est rapide et efficace mais peut cependant augmenter le poids à certaines sondes faibles. Une description de l'algorithme de normalisation par quantile est disponible dans (Bolstad et al., 2003).

Le point commun entre la plupart des approches de normalisation des données de biopuces est qu'elles se basent sur l'hypothèse que la majorité des gènes ont un niveau d'expression invariant entre deux conditions d'hybridation. Cependant, cette hypothèse de stabilité des niveaux d'expression de certains gènes est raisonnable mais généralement pas vérifiables pour certaines expériences et inappropriée pour de nombreuses autres. Les modifications des intensités de signal, faites sur la base de cette hypothèse, peuvent donc engendrer une perte d'information et des erreurs au niveau de l'interprétation des résultats de biopuces. Récemment, Wu and Aryee (2010) ont proposé une nouvelle procédure de normalisation de données en se basant sur un ensemble de sondes de contrôle qui devraient produire des intensités constantes dans les différents échantillons. Ils ont appelé cette procédure SQN (« Subset Quantile Normalization ») afin de la différencier avec la méthode classique de normalisation par quantile qui rend les distributions de l'ensemble de la biopuce égales. En effet, un grand nombre de sondes de contrôle négatif est souvent ajouté à la biopuce pour vérifier la qualité de l'hybridation. Ces sondes sont conçues pour n'avoir aucune complémentarité avec les séquences ciblées. L'utilisation de ces sondes et l'ajout dans l'échantillon biologique des cibles correspondantes en proportion connue (« spike in ») rend alors possible la normalisation des données de biopuces sans faire d'hypothèses sur le comportement des cibles biologiques.

Différentes méthodes de normalisation ont été implémentées dans des programmes tels que ceux écrits en langage R et introduits dans le projet libre « Bioconductor » (www.bioconductor.org): les packages « marray », « limma », « sma », etc. « Bioconductor » est un projet de développement logiciel open source lancé en 2001 et supervisé par une équipe basée principalement au « Fred Hutchinson Cancer Research

Center », et par d'autres membres issus des institutions américaines et internationales. Il est basé principalement sur le langage de programmation statistique R. Sa version est mise à jour deux fois par an. Il contient actuellement (avril 2014) 824 outils dont la plupart sont distribués sous forme de packages R. Il est également disponible en tant que « Amazon Machine Image » (AMI). L'objectif de « Bioconductor » est de fournir à la communauté scientifique (biologistes, bio-informaticiens, statisticiens) des outils pour l'analyse et la compréhension des données génomiques à haut débit. Ces outils sont principalement orientés vers l'analyse de données de biopuces mais proposent aussi d'autres fonctionnalités pour l'analyse des données de séquençage.

3.2. Analyse des données numériques issues des biopuces à ADN

Il existe un grand nombre de méthodes d'analyse des données de biopuce telles que les techniques de classification supervisée ou non, ainsi que de nombreuses méthodes issues d'autres disciplines et adaptées aux données de biopuces à ADN. Le nombre des approches d'analyse proposées dans la littérature est si important que le journal *Bioinformatics* a définit dans un éditorial publié en 2009, certaines normes à respecter pour que les papiers soumis dans le domaine d'analyse des données de biopuces soient sérieusement pris en considération (Rocke et al., 2009). Cet éditorial exige que tout nouvel article apporte une nouvelle méthodologie bien meilleure que les méthodes existantes ou puisse identifier un nouveau défi avec la formulation d'un nouveau problème. L'éditorial exige également une validation importante des approches. Cette validation ne doit pas être réalisée seulement sur des données simulées mais sur des données biologiques réelles.

Les principes de l'analyse des données de biopuces ainsi que les différentes méthodes utilisées ont été décrits dans plusieurs travaux (Quackenbush, 2001; Butte, 2002; Nadon and Shoemaker, 2002; Parmigiani et al., 2003; Leung and Cavalieri, 2003; Hovatta et al., 2005; Barra, 2006; Do and Choi, 2008).

En raison du grand nombre d'approches d'analyse de biopuces existantes, il est difficile d'être exhaustif et seules quelques approches, parmi les plus utilisées, seront présentées dans les paragraphes suivants.

3.2.1. Analyse différentielle

L'intérêt de l'analyse de l'expression différentielle est d'identifier les gènes s'exprimant différemment entre différentes conditions biologiques ou entre un échantillon

de référence et un autre échantillon testé. Ainsi, de nombreuses méthodes statistiques ont été proposées pour l'identification des gènes sur- ou sous-exprimés.

L'une des méthodes connues comme étant la plus simple est celle du « k-fold ». Cette méthode se base sur la valeur du ratio entre les niveaux d'expression relatifs à la condition de référence et ceux des différentes conditions testées. Un gène est considéré comme exprimé de manière différentielle si son ratio « R », appelé « fold change », est supérieur à la valeur « k » du seuil choisi (De Risi et al., 1997). Cependant, la détermination de la bonne valeur de « k » reste la difficulté majeure de cette méthode. En effet, cette valeur dépend du niveau d'expression recherché, de la variabilité d'expression de chaque gène, et de la complexité de l'échantillon hybridé. Dans de nombreuses expériences de biopuces, le niveau d'expression d'un gène est considéré significativement différent si k = 2 (soit R >= 2 pour log2(R) >= 1) pour un gène surexprimé et qu'il est significativement réprimé si k=0.5 (soit R <= 0,5 pour log2(R) <= -1). Néanmoins, ces seuils ne peuvent pas être valides pour toutes les expériences de biopuces à ADN. En effet, à cause de la variabilité du niveau d'expression des gènes, une même valeur seuil peut donner de bons résultats pour un échantillon biologique et un niveau d'expression donné, mais peut aussi engendrer des erreurs d'interprétation dans d'autres cas (Tanaka et al., 2000). Certains biologistes recommandent de calculer un seuil différent pour chaque expérience prenant en compte tous les ratios d'expression des gènes. Cette méthode n'est cependant applicable que si le nombre de gènes ayant une variation d'expression assez élevée est relativement faible par rapport au nombre total de gènes étudiés. Des seuils de confiance peuvent également être déterminés de manière arbitraire pour identifier les variations d'expression significatives.

L'introduction des approches statistiques permet d'estimer la variabilité intra et inter-groupes et de réaliser une analyse plus précise. De nombreux tests statistiques ont ainsi été utilisés pour l'analyse différentielle des gènes tel que la méthode t-test (ou test de student). Dans la méthode t-test, la mesure de la différence entre les moyennes des ratios des niveaux d'expression de la population est étudiée en relation avec l'écart type associé. Le t-test suppose une distribution normale des données qui présentent une variance homogène entre les séries d'observations indépendantes. Une variante du t-test a été proposée par Welch (1938). Elle permet de comparer des groupes dont la variance est hétérogène. Le t-test est dit paramétrique, il n'est applicable que si les données observées ne suivent pas une distribution normale. Cependant, d'autres méthodes non paramétriques ont été développées

pour permettre l'analyse de telles données (Wilcoxon, 1945; Mann and Withney, 1947; Breitling et al., 2004). Ces méthodes sont moins sensibles au mode de distribution des données.

Une alternative à l'inférence statistique classique est l'inférence bayésienne. Efron et al. (2001) ont proposé une approche bayésienne d'identification des gènes différentiellement exprimés appelée EBAM (« Empirical Bayes Analysis of Microarray data »). EBAM se base sur la comparaison de la distribution des différences des niveaux d'expression avec une distribution nulle établie empiriquement.

De nombreuses autres méthodes ont été utilisées pour l'analyse différentielle de l'expression des gènes: SAM (Tusher et al., 2001), test LPE (« Local Pooled Error »), « regularized t-test » (Baldi and Long, 2001), etc.

Outre la détermination des gènes exprimés de manière différentielle, il est souvent important de rechercher et visualiser les éventuels regroupements de gènes ou d'échantillons en fonction de leur profil d'expression en utilisant les méthodes de classification.

3.2.2. Classification

La classification est la technique la plus largement utilisée pour l'analyse des données de biopuces. Les approches de classification consistent à répartir un ensemble d'objets en plusieurs groupes en fonction de leurs similarités. Cette répartition minimise la variabilité intra-groupe et maximise les distances inter-groupes. Pour les biopuces à ADN, cela revient à identifier les groupes de gènes (ou d'échantillons) dont les membres sont similaires mais suffisamment distants des autres groupes, en fonction des expérimentations réalisées. Dans une méthode de classification, chaque gène est considéré comme un individu caractérisé par un ensemble de paramètres dont les valeurs représentent les niveaux d'expression du gène pour chaque échantillon analysé (Slonim, 2002). Les résultats d'intensité sont organisés dans une matrice de niveau d'expression (ou matrice de similarité) dont chaque ligne correspond à un gène et chaque colonne représente un échantillon donné.

Les techniques de classification peuvent être utilisées pour l'analyse de différents types de biopuces: transcriptomiques, taxonomiques ou fonctionnelles. Elles permettent d'évaluer des variations d'expression des gènes chez un organisme donné ou de comparer

des échantillons biologiques complexes en écologie microbienne (biopuces taxonomiques et fonctionnelles).

Il existe principalement deux types d'approches de classification: les approches supervisées et les approches non supervisées.

3.2.2.1. Classification non supervisée

Les approches de classification non supervisée sont des techniques de regroupement (clustering) qui séparent les données observées en groupes distincts sans aucune connaissance préalable des classes existantes. Plusieurs méthodes de classification non supervisée ont été utilisées pour l'analyse des données issues des biopuces. Nous exposons ci-dessous quelques-unes des plus connues.

a. Classification hiérarchique

La classification hiérarchique (Eisen et al., 1998; Spellman et al., 1998) est actuellement la technique la plus utilisée pour l'étude des profils d'expression des gènes. Elle consiste à calculer une matrice de similarité entre les gènes. Un arbre est construit au fur et à mesure pour représenter une hiérarchie des groupes de gènes (figure 13). La liaison entre deux gènes proches correspond à un nœud dans cet arbre, dont les deux gènes sont les feuilles reliées par des branches. La méthode hiérarchique nécessite des calculs de distances de similarité entre les groupes formés. Ces distances peuvent être calculées en se basant sur l'une des nombreuses approches proposées: « Single linkage » (distance minimale entre les éléments des groupes), « Average linkage » (distance moyenne), « Complete linkage » (distance maximale), « Centroid », etc.

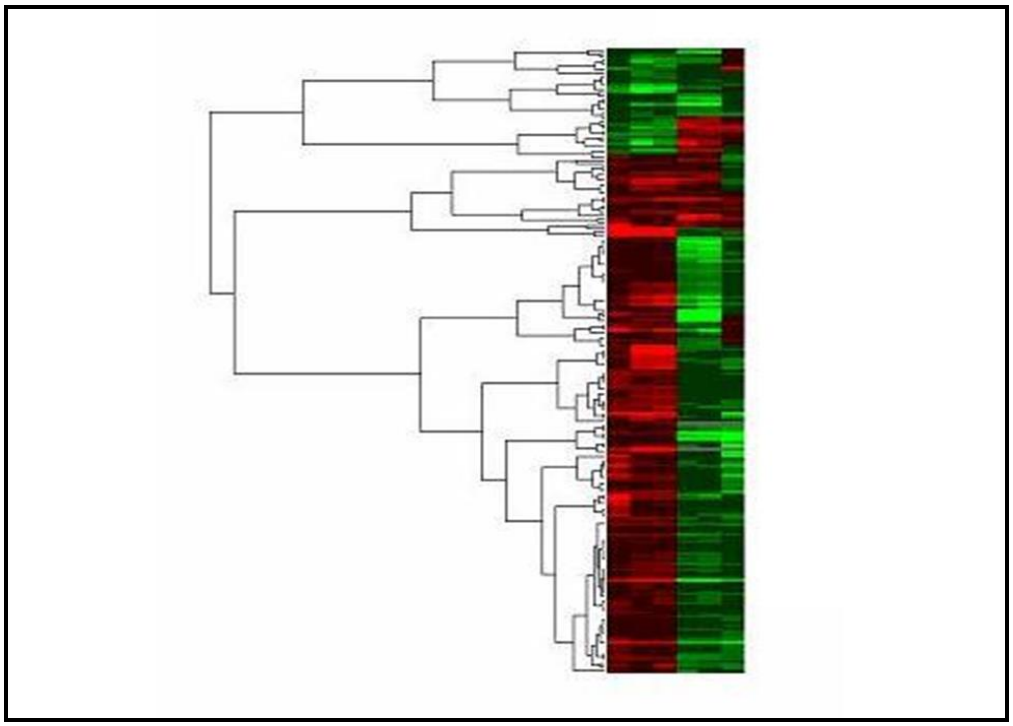

Figure 13. Exemple de représentation de classification hiérarchique d'expression de gènes. Les données sont représentées par une matrice de rectangles colorés dont les colonnes représentent les différentes expériences et les lignes les différents gènes. Les valeurs numériques des ratios d'expression sont représentées sur une échelle de couleurs (rouge: surexprimé et vert: sous-exprimé). A gauche, l'arbre représente l'historique de regroupement des gènes.

L'un des principaux inconvénients de la classification hiérarchique est sa difficulté à traiter de grands jeux de données qui la rend donc coûteuse en ressources mémoires. De plus, le résultat de la classification peut dans certains cas être difficile à interpréter.

La classification hiérarchique a été largement utilisée pour l'analyse des données de biopuces (Moller et al., 2008; Savage et al., 2009; Claesson et al., 2009; Grossi et al., 2010; Mackay et al., 2011; Rajan et al., 2013).

b. Regroupement en nuées dynamiques

Le regroupement en nuées dynamiques ou « k-means clustering » consiste à calculer pour chaque groupe un point central (sa moyenne), et d'attribuer chaque gène au cluster le plus proche. La distance entre un gène et un cluster est définie comme étant la distance entre le gène et la moyenne du cluster. Les étapes du « k-means » sont les suivantes:

(1) Choisir arbitrairement K points qui seront définis comme les valeurs initiales des centres des k clusters (initialisation stochastique).

(2) Affecter chaque gène G au cluster Ci de centre Mi tel que la distance entre G et Mi est minimale.

(3) Recalculer le centre Mi de chaque cluster.

(4) Répéter les étapes 2 et 3 si une affectation est réalisée.

La popularité de l'algorithme k-means vient de sa simplicité, sa rapidité et sa capacité de traiter des jeux de données importants. Cependant, dans cette méthode, le nombre de groupes « k » doit être défini à l'avance. Cela nécessite un travail préalable pour choisir le nombre de groupes optimal, ce qui n'est pas toujours facile et possible dans le contexte de l'analyse des résultats de biopuces à ADN. Les résultats finaux sont fortement dépendants de ce choix du nombre de clusters. Toutefois, l'utilisateur peut essayer différents nombres de cluster afin de définir celui qui correspond le mieux aux données.

L'algorithme k-means a été utilisé dans plusieurs études pour l'analyse des données issues des expériences d'hybridation de biopuces à ADN (Cao et al., 2008; Tothill et al, 2008; Heintzman et al., 2009).

c. Cartes de Kohonen

Un des modèles de réseau de neurones les plus populaires mais également les plus utilisés en analyse de données d'expression de gènes est le SOM « Self Organizing Maps » souvent appelé cartes de Kohonen (Kohonen, 1982; Kohonen, 2001). Le SOM est un réseau non supervisé d'apprentissage pour le regroupement et la visualisation des données de grande dimension. Le principe de base du SOM est proche de celui de la méthode k-means mais diffère au niveau de la réactualisation des positions des centres de classes. L'idée est d'introduire un espace de représentation de dimension réduite (une ou deux dimensions), représentant les relations de voisinage entre classes.

L'utilisation des cartes de Kohonen, bien qu'étant une approche non supervisée, exige la prédéfinition de certains paramètres pour son initialisation tels que la structure topologique de la carte et le nombre de nœuds (ou neurones).

Cette méthode a été par exemple utilisée pour l'analyse et la visualisation des profils d'expression de gènes de levure (Törönen et al., 1999) et humains (Herrero et al., 2001, Hsu et al., 2003, Bédrine-Ferran et al., 2004).

L'avantage de cette technique est sa capacité à préserver les relations topologiques entre l'espace d'entrée et l'espace de sortie. En effet, contrairement à la classification hiérarchique et à l'algorithme k-means, la position des groupes sur la carte reflète le niveau de similarité entre les données. Ainsi, la carte montre une certaine organisation et il est possible d'évaluer la structuration des données analysées par la simple observation de la carte. Les algorithmes SOM sont, de plus, peu sensibles aux données manquantes (Cottrell and Letrémy, 2005).

Cependant, la complexité de la phase d'initialisation de ces algorithmes, en raison du besoin de prédéterminer la structure et la taille de la carte, représente une limitation significative de l'approche SOM. En effet, le choix des meilleures valeurs pour ces paramètres n'est pas simple. Souvent, à la fin de l'analyse, on constate qu'une carte différente aurait été plus appropriée. De ce fait les valeurs sont généralement définies après plusieurs simulations successives sur des cartes de différentes tailles.

3.2.2.2. Classification supervisée

Les approches de classification supervisée comprennent une phase d'apprentissage permettant d'établir des règles de classification à partir d'un jeu de données dont la classification est connue. Ces règles sont ensuite généralisées et utilisées pour prédire la classification de nouvelles données ne faisant pas partie du jeu de données initial.

Comme pour les approches non supervisées, plusieurs techniques de classification supervisées existent. Seules deux des approches les plus connues et actuellement les plus utilisées seront présentées.

a. « Support Vector Machines »

Les techniques des « Support Vector Machines » (SVM) ou machine à vecteur de support (Vapnik, 1998) se basent sur la séparation des données d'un ensemble de tests par un hyperplan maximisant la distance aux points de test. Si aucune séparation par un tel hyperplan n'est possible, une coopération entre SVM et une technique de noyaux pour une séparation non linéaire est possible. Les données dont on ne connaît pas l'étiquetage sont ensuite classées par rapport à l'hyperplan séparateur (avec un intervalle de confiance dépendant par exemple de la distance à l'hyperplan).

Les techniques SVM ont été largement utilisées pour classer les données de biopuces (Mukherjee, 1999; Furey et al., 2000; Guyon et al., 2002; Valentini, 2002; Lee and Lee, 2003; Mukherjee, 2003; Huang and Chang, 2007; Mukhopadhyay, 2009; Debnath and Kurita, 2010; Nanni and Lumini, 2011). Par exemple, Brown et al. (1997) ont utilisé les SVM sur des données d'expression de gènes de la levure *Saccharomyces cerevisiae* et ont comparé ces techniques avec d'autres méthodes supervisées. Pavlidis et al. (2002) ont de leur côté appliqué les SVM sur des données hétérogènes: données d'expression et profil génétique de chaque gène.

b. « K Nearest Neighbor »

La technique des k plus proches voisins ou KNN (« **K** **N**earest **N**eighbor ») est une méthode de classification intuitive et très simple qui consiste à classer les nouvelles données non étiquetées (non classées) sur la base de leur similarité avec les données de la base d'apprentissage (figure 14). En effet, pour un élément *X* non étiqueté, on choisit la classe qui a le plus grand nombre d'éléments, parmi les k éléments étiquetés (appartenant à la base d'apprentissage), proches (plus similaires) de *X*.

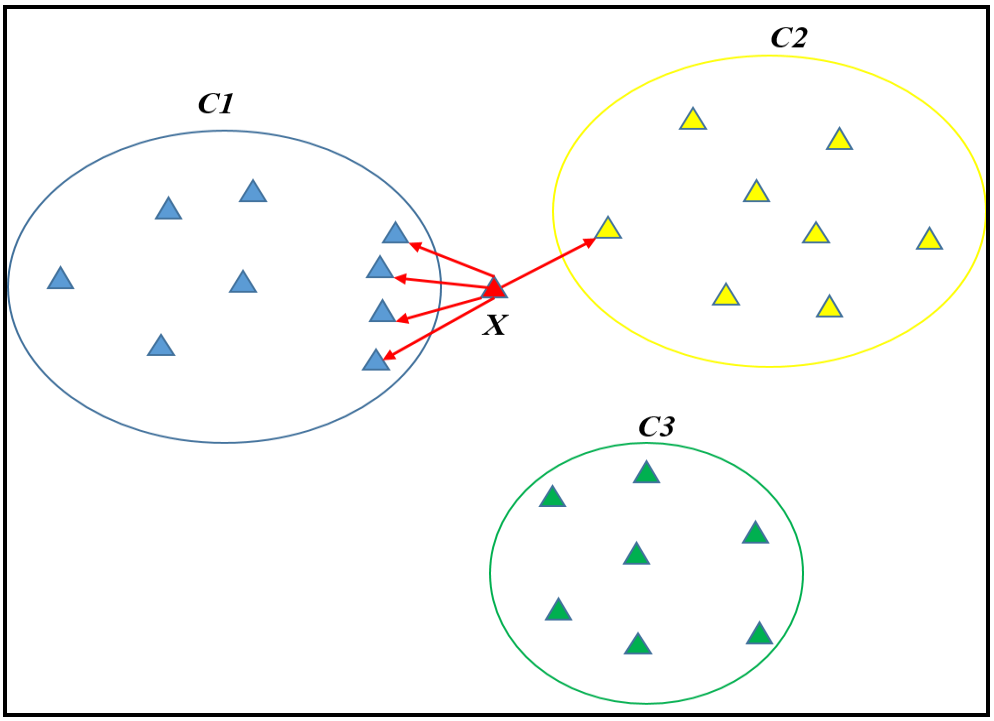

Figure 14. Exemple d'utilisation de l'algorithme des k plus proches voisins: dans cet exemple, on a 3 classes. On utilise la distance euclidienne et on considère un nombre de voisins k = 5. Parmi les 5 plus proches voisins du nouvel élément *x*, 4 appartiennent à la classe *C1* et 1 à *C2*. L'élément *x* est donc affecté à la classe la plus représentée *C1*.

Trois paramètres sont à nécessaires pour appliquer l'algorithme KNN: le nombre des plus proches voisins recherchés (k), le seuil de décision (marge d'erreur) et la métrique de distance utilisée.

La méthode KNN a été largement utilisée pour classer des données de biopuces (Lu et al., 2005; Miller et al., 2005; Hahn et al., 2008; Hoshida et al., 2008; Rosenfeld et al., 2008).

Cette technique est très simple mais elle est sensible aux valeurs aberrantes. Les résultats peuvent également être différents en fonction de la valeur de k choisie.

3.2.3. Analyse en composante principale

Le but d'une Analyse en Composantes Principales ACP (Dunteman, 1989) est de résumer et de hiérarchiser l'information contenue dans la matrice des niveaux d'expression des gènes. Il s'agit d'un changement de coordonnées dans l'espace permettant de représenter plus synthétiquement les variations des données, sans les dénaturer.

L'analyse en composante principale consiste en les étapes suivantes:

(1) Centrer et réduire les données afin de les standardiser.

(2) Créer la matrice de corrélation entre les variables: matrice carrée symétrique d'ordre p (p est le nombre de colonnes de la matrice qui correspond au nombre d'échantillons hybridés).

(3) Trouver les vecteurs propres de la matrice de corrélation ainsi que leur valeur propre associée.

(4) Calculer les coordonnées des gènes et des échantillons sur les vecteurs propres pour la représentation graphique. Les gènes sont projetés sur les axes propres principaux donnant accès à une représentation 2D ou 3D des variations des données initiales.

L'ACP a été largement appliquée à l'analyse des résultats de biopuces (D'Haeseleer et al., 2000; Raychaudhuri et al., 2000; Crescenzi and Giuliani, 2001; Yeung and Ruzzo, 2001).

3.2.4. Autres méthodes

De nombreuses autres approches issues d'autres disciplines scientifiques ont été utilisées pour analyser les données issues des biopuces à ADN. Parmi elles, on peut citer les méthodes basées sur la logique floue (Woolf and Wang, 2000; Azuaje, 2001) ou la théorie des graphes (Ben-Dor et al., 1999; Hartuv et al., 1999; Xing and Karp, 2001). On peut aussi trouver d'autres méthodes issues des domaines de l'intelligence artificielle et des bases de données (Lemkin et al., 2000; Hanczar et al., 2003, 2007; Gutkin et al., 2009), de la théorie de l'information (Fuhrman et al., 2000), etc.

3.3. Logiciels d'analyse des données de biopuces

La diversité des types et applications des biopuces à ADN, ainsi que leur grande popularité ont conduit au développement d'un nombre important d'outils informatiques pour aider à bien analyser et interpréter les données issues des hybridations de biopuces. Les outils les plus populaires sont décrits dans (Dudoit et al., 2003; Mehta and Rani, 2011; Koschmieder et al., 2012). En raison du grand nombre de logiciels d'analyse de biopuces existants, nous présentons ci-dessous seulement certains d'entre eux parmi les plus connus.

3.3.1. TM4

TM4 (Saeed et al., 2006) est un ensemble complet de programmes Java autonomes pour l'étude d'expression des gènes. Tous ces programmes, ainsi que leurs codes sources, sont publiquement et librement disponibles à partir du site web http://www.tm4.org/. TM4 permet de collecter, gérer et analyser les données issues des expériences de biopuces à ADN. Il comprend quatre applications principales: MADAM « MicroArray Data Manager » pour sauvegarder des données de biopuces deux couleurs dans une base de données relationnelle, « Spotfinder » pour analyser les images de biopuces et générer les données numériques, MIDAS « Microarray Data Analysis System » pour filtrer et normaliser les données numériques fournies par Spotfinder et pour les préparer à l'analyse et l'interprétation et MeV « Multiexperiment Viewer » pour visualiser et d'analyser les données dans TM4. Cette dernière application permet ainsi de réaliser différents types d'analyses telles que la classification, le clustering ou des tests statistiques avec une interface graphique assurant la navigation entre les différents résultats obtenus.

3.3.2. SAM « Significance Analysis of Microarrays »

SAM est un package R et une macro Excel distribués par l'Université de Stanford. C'est un outil statistique qui permet de vérifier si des différences d'expression de gènes sont statistiquement significatives. Le logiciel SAM est basé sur la méthode SAM présentée dans (Tusher et al., 2001). Cette méthode permet de trouver les gènes différentiellement exprimés entre différentes expérimentations en se basant sur un t-test modifié et des permutations aléatoires des données d'expression. Un score est attribué à chaque gène sur la base du changement de son niveau d'expression par rapport à l'écart type des mesures répétées de ce gène. Les différences d'expression des gènes ayant des scores supérieurs à un seuil donné sont considérées comme potentiellement significatives. SAM calcule aussi un taux de faux positifs FDR (« False Discovery Rate »).

L'outil SAM peut être gratuitement téléchargé sur http://www-stat.stanford.edu/~tibs/SAM/ pour les utilisateurs académiques et non-académiques après une étape obligatoire d'enregistrement. La dernière version actuellement disponible est SAM4.01 (release décembre 2013). Cet outil supporte plusieurs formats d'analyse: « Quantitative », « One class », « Two class », « Paired », « Multiclass », « Survival data », « Time course », et « Pattern discovery » (Chu et al., http://www-stat.stanford.edu/~tibs/SAM/sam.pdf).

3.3.3. Genesis

Genesis (Sturn et al., 2002) est un package d'outils Java pour le traitement et l'analyse de grands ensembles de données issues des biopuces à ADN. Il contient plusieurs modules pour la transformation, la normalisation et la représentation graphique des données de biopuces. Il propose également différentes méthodes pour la classification des données d'expression des gènes: classification hiérarchique, k-means, SOM (« self organizing maps »), analyse en composantes principales et machine à vecteur de support SVM.

La dernière version de ce logiciel est Genesis 1.7.6 (release septembre 2010). Genesis est disponible librement pour toute utilisation non commerciale (académique ou gouvernementale). Il peut être gratuitement téléchargé sur http://genome.tugraz.at/genesisclient/genesisclient_download.shtml. Une application web pour l'analyse des résultats de biopuces à grande échelle, basée sur les outils de Genesis, est également disponible sur https://carmaweb.genome.tugraz.at/genesis/.

3.3.4. Chipster

Chipster (Kallio et al., 2011) est un logiciel gratuit et open source d'analyse de données à haut débit contenant une collection de 280 outils d'analyse pour les données de séquençage NGS et des biopuces à ADN de type Affymetrix (GeneChips, ExonChips and SNP), Illumina et Agilent. Chipster est développé en Java, il est compatible avec Windows, Linux et Mac OS. Il est basé sur une architecture client-serveur (figure 15). Les données sont importées et visualisées du côté client, alors que les calculs et traitements sont réalisés côté serveur. Chipster peut être gratuitement téléchargé sur http://chipster.sourceforge.net/downloads.shtml.

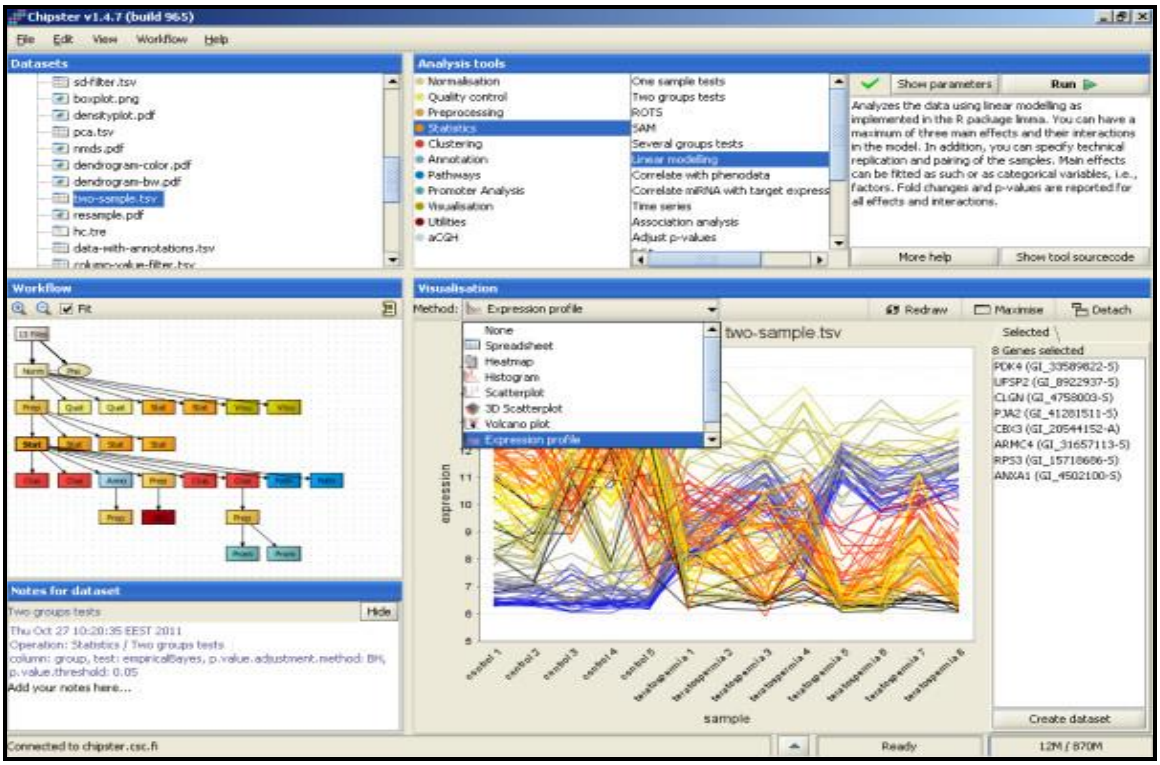

Figure 15. Interface client de Chipster (Source: Kallio et al., 2011).

Chipster propose une grande variété de techniques pour la normalisation et la correction du bruit de fond (RMA, GCRMA, loess, quantile, VSN, mas5, normexp, etc.), le contrôle de qualité (RLE plot, NUSE plot, MA plot, density plot, etc.), l'analyse différentielle (t-test, SAM, ANOVA, F-test, Fold change, etc.), le clustering (K-means, classification hiérarchique, SOM Self-organizing map) et d'autres applications pour l'analyse des données de biopuces. Chipster offre également la possibilité d'importer ou d'exporter des données à partir des bases de données publiques comme « GEO »

et « ArrayExpress ». Il est possible d'ajouter de nouveaux outils dans Chipster. Le principe est simple: écrire une courte description des entrées, sorties et paramètres de l'outil à l'aide d'un outil de notation. Ensuite, copier cette description ainsi que le code source de l'outil dans un répertoire spécifique du service de calcul, ou bien ajouter simplement une référence au fichier binaire de l'outil dans le fichier de configuration Chipster s'il s'agit d'un outil en ligne de commande.

3.3.5. PhyloTrac

PhyloTrac (Schatz et al., 2010) est un outil informatique spécifique à l'analyse phylogénétique et la visualisation des données de biopuces PhyloChip. PhyloTrac est disponible gratuitement pour l'utilisation académique, et fonctionne sur Windows, Linux et Mac OS. Il peut être téléchargé sur http://www.phylotrac.org/Download.html. PhyloTrac permet une analyse des échantillons environnementaux des biopuces PhyloChip à travers une variété de modèles de visualisations intégrés (affichage des intensités de sondes, arbre phylogénétique, graphe « heatmap », graphe de coordonnées parallèles, feuilles de calcul textuelles, etc.), ainsi que la quantification des groupes taxonomiques OTUs (« Operational Taxonomic Units »), la normalisation des intensités de sondes et le clustering (classification hiérarchique). Utilisé avec une PhyloChip, PhyloTrac permet l'identification et l'analyse des communautés procaryotes dans des échantillons environnementaux complexes.

L'analyse PhyloTrac prend comme entrée un fichier de format CEL (« Affymetrix probe results file ») contenant les résultats d'hybridation de la PhyloChip. Il réalise d'abord la normalisation des données en utilisant les méthodes décrites dans (DeSantis et al., 2007). PhyloTrac soustraire le bruit de fond des données et les normaliser en fonction des valeurs de bruit de fond et des intensités des sondes de synthèse. Ensuite, les intensités d'hybridation de chaque sonde PM (« perfect-match ») sont comparées à celles de la sonde MM (« mismatch probe ») correspondante (sonde générant un mésappariement en position centrale avec la séquence ciblée) pour estimer les hybridations spécifiques. Un OTU est considéré comme présent dans l'échantillon lorsque la fraction positive (PF) des sondes qui le ciblent est supérieure à un seuil donné par l'utilisateur (le seuil par défaut est égal à 95%). Un score d'hybridation est aussi calculé pour chaque OTU présent. Ce score correspond à l'intensité moyenne de ses sondes PM. Ces résultats seront ensuite enregistrés et utilisés pour réaliser les représentations graphiques et le clustering hiérarchique (figure 16).

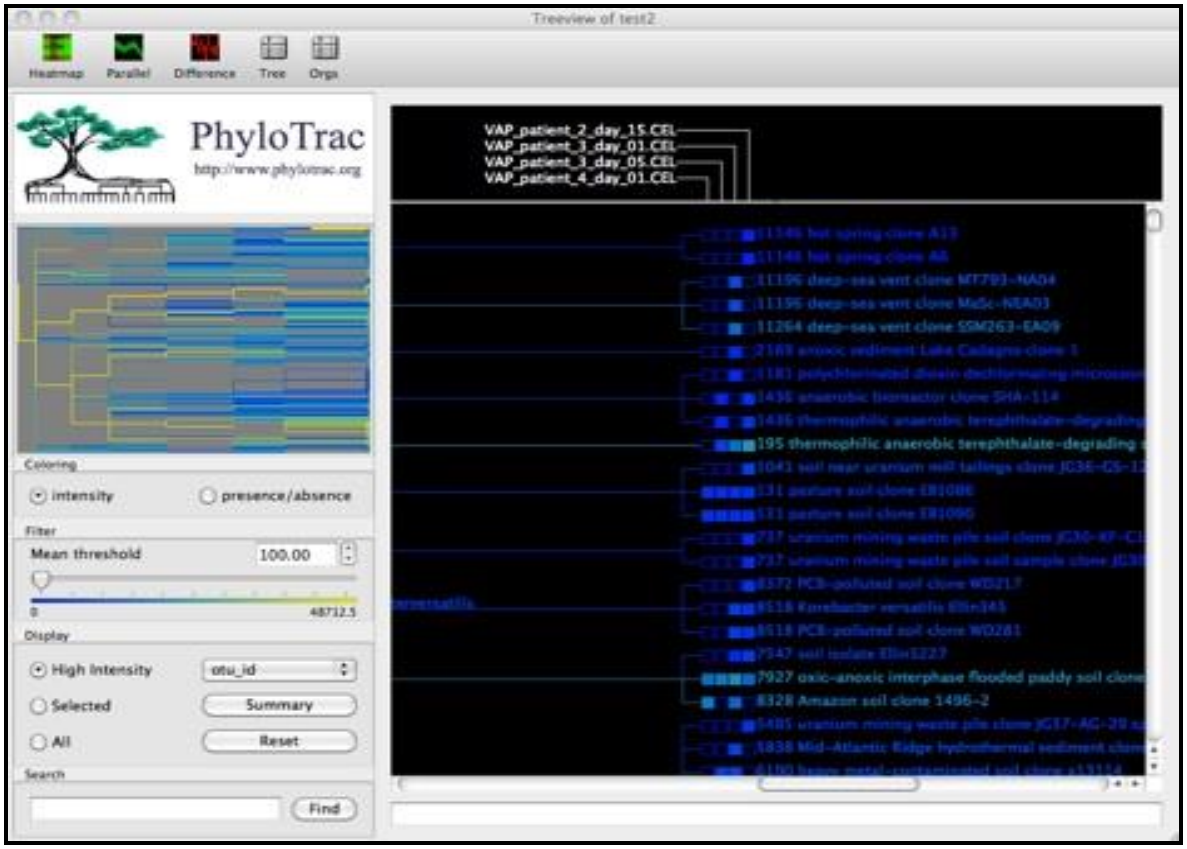

Figure 16. Interface graphique de PhyloTrac

(Source: www.phylotrac.org/Getting_Started.html, mars 2014).

4. Conclusion

Dans ce chapitre, nous nous sommes intéressés aux problèmes de développement et d'analyse des biopuces à ADN en écologie microbienne. En effet, de nombreuses approches pour la sélection de sondes oligonucléotidiques ont été proposées pour les différentes applications de biopuces. Ces approches sélectionnent souvent des sondes ciblant seulement des séquences connues et déjà répertoriées dans les bases de données internationales. Cependant, il est largement admis que la grande majorité des séquences des microorganismes habitants notre planète n'a pas encore été découverte. De ce fait, les biopuces utilisant des sondes exploratoires c'est-à-dire ciblant des organismes connus et encore inconnus représentent une piste prometteuse à développer. Mais l'élaboration de telles sondes est une tâche loin d'être facile. En effet, un outil de détermination de sondes exploratoires doit permettre d'utiliser l'énorme masse de données de séquences présentes dans les bases de données publiques et sélectionner des dizaines de milliers de sondes à la

fois. De plus, l'utilisation d'un tel nombre de sondes génère une quantité considérable de résultats à traiter.

Pour l'étape de normalisation des données de biopuces, plusieurs techniques sont adaptées et peuvent être utilisées pour la plupart des applications et types de biopuces. Cependant, ces techniques sont souvent basées sur une hypothèse de stabilité des intensités d'hybridation de la majorité des séquences ciblées entre les différents échantillons hybridés. Une hypothèse qui reste généralement peu vérifiable ou irréaliste pour de nombreuses expériences.

En ce qui concerne l'interprétation et l'analyse des données de biopuces après normalisation, les outils développés sont souvent spécifiques à certaines applications (biopuces transcriptomiques) et types de biopuces (surtout aux biopuces Affymetrix). Cependant, les méthodes de classification développées pour évaluer des variations d'expression des gènes chez un seul organisme sont utilisables pour comparer des échantillons biologiques complexes en écologie microbienne (analyse des biopuces taxonomiques et fonctionnelles). De plus, comme la plupart des premiers logiciels n'étaient pas adaptés à l'exploitation des résultats beaucoup plus complexes produits par les biopuces utilisées en écologie microbienne, des outils d'analyse phylogénétique dédiés tels que PhyloTrac (Schatz et al., 2010) pour les biopuces PhyloChip ont été développé.

La complexité des outils de développement et d'analyse des biopuces est en constante augmentation et de ce fait les besoins en temps de calcul dépassent largement la capacité d'un simple ordinateur. Nous nous intéressons donc par la suite aux outils du calcul intensif ainsi qu'aux meilleurs approches logicielles pour réaliser les outils nécessaire à la conception, la gestion et l'analyse des biopuces.

Chapitre 3: Etat de l'art: Architectures de calcul intensif et concepts d'ingénierie dirigée par les modèles

1. Introduction

Les techniques et architectures de calcul intensif, disponibles de nos jours, offrent des capacités de calcul et de sauvegarde de données très importantes pouvant réduire le temps de calcul et la complexité du problème de conception et d'analyse des biopuces à ADN. En outre, l'exploitation des architectures de calcul intensif peut être combinée avec l'utilisation des nouvelles approches du génie logiciel inspirées de la modélisation, notamment l'ingénierie dirigée par les modèles (IDM), afin d'améliorer l'efficacité des approches de sélection de sondes exploratoires.

Dans ce chapitre, nous exposerons les principes du calcul intensif et ses principales architectures actuellement disponibles. Ensuite, la deuxième partie de ce chapitre décrira les concepts de l'ingénierie dirigée par les modèles.

2. Le calcul intensif

Avec l'augmentation continue de la quantité de données produite dans le domaine de la biologie et disponible dans les banques de données de séquences, les algorithmes bioinformatiques appliquées à ces données deviennent de plus en plus complexes et leurs temps de calcul peuvent rapidement dépasser la capacité d'une seule machine (Zomaya, 2006). L'intérêt de l'utilisation du calcul intensif pour la bioinformatique est d'améliorer les performances des applications et les rendre capables de s'exécuter en un temps raisonnable. L'objectif est de gagner en efficacité et en précision, mais aussi de faire face à l'augmentation exponentielle des données à traiter. Dans le cadre de la détermination de sondes pour les biopuces à ADN, la recherche de sondes spécifiques et exploratoires nécessite de faire appel à des programmes de comparaison de séquences pour tester un grand nombre de sondes contre une large base de données de séquences. Les applications de détermination de sondes nécessitent donc des ressources de calculs croissantes (Zhu et al., 2006; Simmler et al., 2003). Les architectures de calcul intensif représentent un outil de choix pour répondre à ces besoins, surtout que la capacité et la disponibilité des ressources du calcul intensif sont actuellement en croissance continue. En effet, depuis une quinzaine d'années, le marché du calcul haute performance (HPC) enregistre une croissance importante d'environ 7% en moyenne annuellement, et ce malgré les crises. Ce marché a atteint un total de plus de 11 milliards de dollars dans le monde en 2012 et 500 millions d'euros en France. Les leaders du domaine qui détiennent à eux seuls 74% de

ce marché en 2012 sont IBM, HP et Dell. Le dynamisme et la croissance du marché de calcul intensif vient de la continuité de la demande d'équipement dans le secteur public qui représente actuellement 70% du marché. Les investissements se justifient par l'apport considérable du calcul intensif sur la science (augmentation des performances et réduction des durées de traitement) et sur l'économie (création d'emplois). Ainsi, les ressources de calcul des laboratoires de recherches et des centres de calcul académiques, dans les pays leaders dans ce domaine en termes d'investissements, ne cessent de croître. En France, l'importance du calcul intensif, a poussé le ministère de la recherche à créer, en 2006, la société civile GENCI[1] pour la gestion des moyens de calcul pour la recherche. GENCI est détenue à 49% par l'Etat, 20 % par le CEA, 20 % par le CNRS, 10% par les universités et 1% par l'Inria. Elle a comme objectif de placer la France au meilleur niveau européen et international dans le domaine du calcul intensif. Un budget de 35 M€ a été alloué à GENCI en 2012. En parallèle, le ministère a créé, dans la même année 2007, le « Comité Stratégique pour le Calcul Intensif » (CSCI) chargé de suivre les activités nationales et européennes dans le domaine du calcul intensif et de formuler des propositions sur l'organisation, le renouvellement et l'utilisation des ressources de calcul intensif.

Cependant, le secteur public n'est pas le seul à investir en HPC. Le secteur privé a aussi sa part du marché. En effet, les entreprises, dans plusieurs domaines (transport, énergie, pharmacie, finance, etc.), participent de plus en plus à l'émergence du HPC en se dirigeant vers l'utilisation du calcul intensif comme outil indispensable pour l'innovation, le développement et la compétitivité. Un outil qui permet de réduire les durées et les coûts des développements et de gagner en qualité et en précision. Comme exemples d'entreprises qui investissent de plus en plus en HPC, on peut citer Boeing, Airbus, et « Total » qui vient d'acquérir, en 2012, un supercalculateur de plus de 110.000 cœurs de calcul. La pénétration du HPC reste néanmoins faible dans l'industrie française, en particulier pour les PME, selon le rapport final du « Comité Stratégique du Calcul Intensif » publié en juin 2013.

L'émergence des ressources de calcul intensif, reste fortement encouragée par l'importante réduction de leurs coûts d'acquisition qui rend cette technologie de plus en plus abordable. De plus, la capacité d'usage et d'exploitation ne cesse de croître, surtout avec les nouvelles compétences humaines qui ont profité des formations universitaires de

[1] http://www.genci.fr

plus en plus répandues en calcul intensif, que ce soit au niveau ingénieur, master ou doctorat (pour la France, le site http://calcul.cnrs.fr/ onglet formations recense les formations disponibles). Le nombre de ces formations reste cependant insuffisant en France selon le rapport final CSCI 2013 qui a recommandé la création de plus de formations de master et de projets de doctorat conjoints entre universités et composantes de calcul.

Avec l'émergence des ressources de calcul intensif et la formation de nouvelles compétences humaines capables de gérer la complexité de l'utilisation d'architectures de calcul haute performance en évolution continue, il reste une question importante à poser avant l'acquisition de nouvelles ressources ou le développement de nouvelles applications: quelle architecture ou outil de calcul intensif utiliser en fonction de ses besoins applicatifs?

2.1. Classification de Flynn

Plusieurs approches de classification des architectures d'ordinateurs ont été proposées (la classification de Flynn (1966), de Kuck (1978), de Treleaven (1982), de Hwang and Briggs (1984), de Skillicorn (1988), de Hockney and Jesshope (1988), de Bell (1992), etc.). La plupart de ces classifications représente une modification ou une extension de la classification de Flynn qui est l'approche la plus ancienne et la plus connue. C'est une approche proposée par l'américain « Michael J. Flynn » dans les années 60 (Flynn, 1966). Elle classe les différentes architectures en se basant sur deux flux: le flux d'instructions et le flux de données. Ces flux peuvent être simples ou multiples, et leurs combinaisons forment ainsi 4 catégories de machines (Flynn, 1972):

- ✓ **SISD (« Single Instruction Single Data Stream »)**: un seul flux d'instructions est exécuté sur un seul flux de données. C'est le cas des anciennes machines monoprocesseur (séquentielles) qui ne peuvent offrir aucun type de parallélisme, ni au niveau des instructions ni au niveau des données. Cela correspond à la machine de Von-Neumann (figure 17) et aux anciens microprocesseurs datant d'un peu plus d'une dizaine d'années.

- ✓ **SIMD (« Single Instruction Multiple Data Stream »)**: c'est une machine qui introduit le parallélisme au niveau des données: on parle en général de parallélisme de données. Pour un système de type SIMD, le même flux d'instructions est

appliqué à différents ensembles de données. C'est le cas des processeurs vectoriels ou des processeurs graphiques.

- ✓ **MISD (« Multiple Instruction Single Data Stream »)**: c'est un système dans lequel différents flux d'instructions sont appliqués à un seul flux de données. Dans la pratique, ce type d'architecture n'est pas utilisé sauf dans les systèmes de tolérance aux pannes.

- ✓ **MIMD (« Multiple Instruction Multiple Data Stream »)**: c'est un ordinateur qui est composé de plusieurs instructions indépendantes s'exécutant simultanément sur plusieurs données indépendantes. C'est le cas des architectures distribuées comme les SMP (« Symmetric MultiProcessors »).

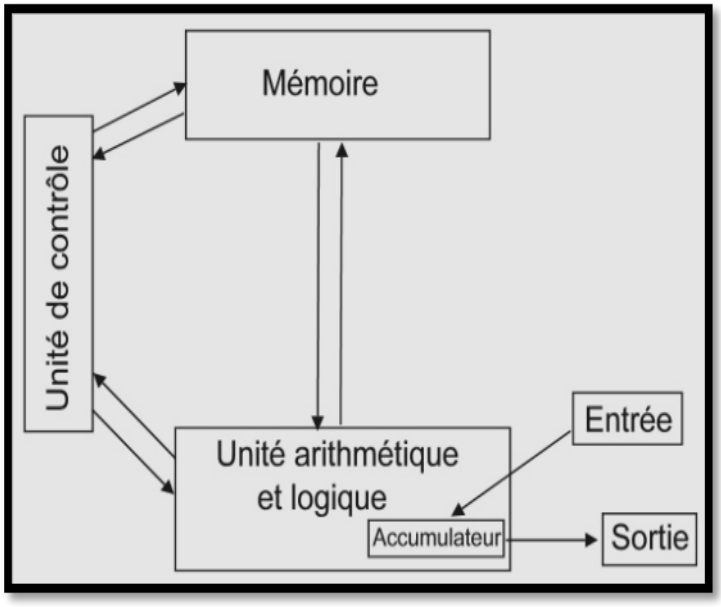

Figure 17. Machine de Von-Neumann. Source Wikipedia traduction de l'anglais.

2.2. Classification par type de mémoire pour les systèmes MIMD

Trois classes de machines peuvent être identifiées en fonction de la gestion de la mémoire pour les systèmes MIMD:

- ✓ **Machines à mémoire partagée (« Shared Memory » SM)**: c'est une machine dont la mémoire peut être accédée simultanément par plusieurs processus. Il s'agit d'une zone de la mémoire vive (RAM pour « Random Access Memory ») partagée

par tous les processeurs. On distingue trois types de mémoire partagée: UMA (« Uniform Memory Access »), NUMA (« Non Uniform Memory Access ») et COMA (« Cache Only Memory Access »).

- ✓ **Machine à mémoire distribuée (« Distributed Memory » DM)**: c'est une machine dont chaque processeur possède sa propre mémoire à laquelle il peut accéder pour exécuter son propre flux d'instructions. Chaque processeur est donc indépendant et constitue avec sa propre mémoire un nœud. Un réseau dédié relie les différents nœuds. L'accès à une donnée située sur un autre nœud doit se faire par un mécanisme d'échange de messages.

- ✓ **Machine à mémoire virtuelle partagée (« Virtual Shared Memory » VSM)**: c'est une machine dont la mémoire est physiquement distribuée mais chaque processeur peut accéder aux mémoires spécifiques des autres processeurs.

La figure 18 résume les différentes architectures possibles en fonction de la classification de Flynn et par type de mémoire.

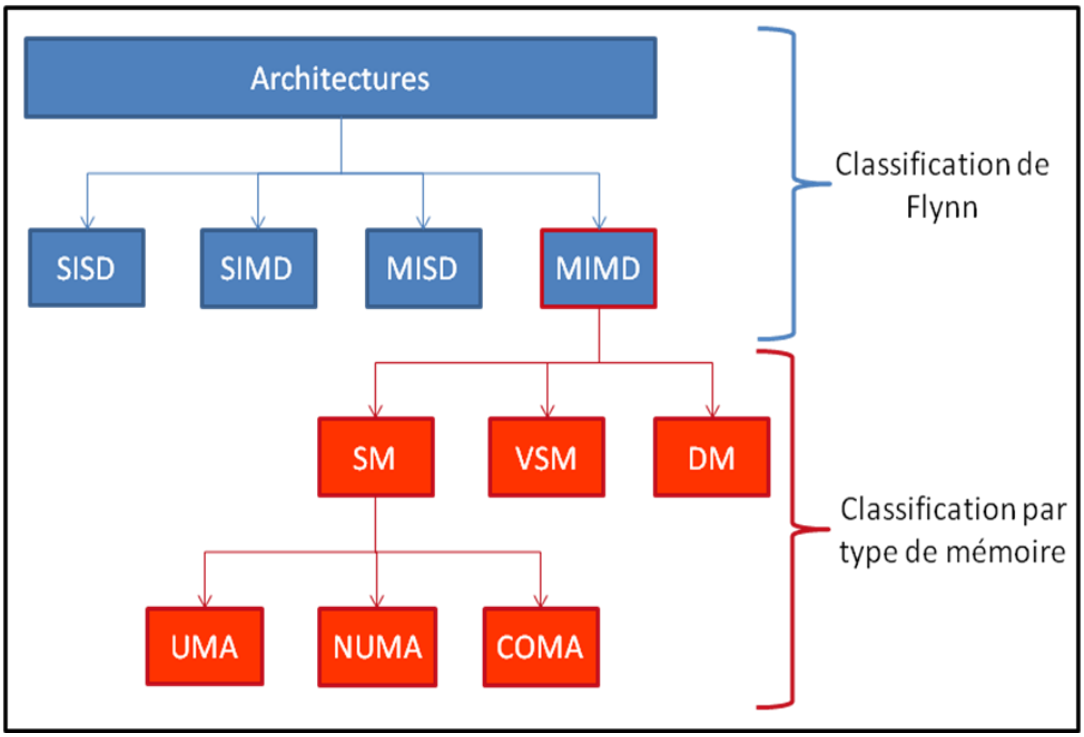

Figure 18. Classification des architectures des machines.
Les acronymes sont donnés dans les paragraphes 4.1 et 4.2.

2.3. Supercalculateurs

Les supercalculateurs sont des systèmes très puissants conçus pour réaliser une très haute performance en termes de puissance de calcul, par rapport aux autres ordinateurs d'une époque donnée. Ils sont dédiés à l'exécution de tâches très complexes de calcul intensif telles que les modélisations météorologiques, les simulations physiques, les modélisations moléculaires, etc.

Face aux besoins toujours croissants des utilisateurs, plusieurs architectures ont été développées depuis l'apparition des premiers supercalculateurs avec l'IBM 7030, aussi appelé Stretch, en 1961. L'IBM 7030 est le premier ordinateur avec un processeur doté d'une architecture pipeline. Le pipeline est une technique permettant d'exécuter plusieurs instructions en même temps sur un processeur. Avec un pipeline, le processeur peut commencer une nouvelle instruction avant que l'exécution de la précédente ne soit terminée. Ainsi, plusieurs instructions se trouvent simultanément en cours d'exécution sur le processeur. Afin de mettre en œuvre un pipeline, l'exécution des instructions est découpée en plusieurs étapes (sous parties élémentaires). Chaque étape est effectuée dans un étage du pipeline. Par exemple, un processeur avec un pipeline RISC classique (« Classic RISC pipeline »), est composé de 5 étages:

(1) **IF** (« Instruction Fecth »): chargement de l'instruction à exécuter,

(2) **ID** (« Instruction Decode »): décodage de l'instruction et adressage des registres,

(3) **EX** (« Execute »): exécution de l'instruction,

(4) **MEM** (« Memory »): accès mémoire,

(5) **WB** (« Write Back »): écriture du résultat dans un registre.

Un processeur superscalaire dispose de plusieurs pipelines en parallèle afin d'exécuter plusieurs instructions simultanément parmi une suite d'instructions. Le premier supercalculateur équipé d'un processeur superscalaire était le CDC 6600 (1964), de la société « Control Data Corporation CDC ». Il est 3 fois plus rapide que l'IBM 7030 et a été l'ordinateur le plus puissant entre 1964 et 1969, date d'apparition de son successeur le CDC 7600.

À partir des années 1970, Seymour Cray, qui était le principal concepteur des supercalculateurs chez Control Data, quitte son entreprise pour créer sa propre société en préconisant l'utilisation d'architectures vectorielles. Cette approche de parallélisation permet d'exécuter une même instruction sur plusieurs fragments de données simultanément afin d'améliorer les performances des supercalculateurs. Les premiers supercalculateurs vectoriels ont été conçus par « Texas Instruments TI» (le ASC « Advanced Scientific Computer » entre 1966 et 1973), par Control Data (le CDC STAR-100 en 1974) et le plus célèbre a été conçu par « Cray Research Inc » (le Cray-1 en 1976). Ce dernier Cray-1 possédait une puissance de calcul de 138 MégaFLOPS[2]. Très connu dans l'histoire des supercalculateurs, ce type d'architecture était la plus aboutie et constituait la référence de l'époque. En 1982 le Cray X-MP est le premier supercalculateur parallèle à multiprocesseur vectoriel inventé par « Steve Chen ». Le Cray X-MP qui a succédé au Cray-1, possédait alors une performance de 800 MégaFlops. Il a été ensuite suivi, en 1985, par Cray-2, toujours développé par Seymour Cray. Le Cray-2, composé de 2 à 4 processeurs, fut le premier supercalculateur à dépasser le GigaFLOPS avec une puissance pouvant atteindre 1,9 GigaFLOPS. Cray Research a ensuite introduit le Cray Y-MP en 1988. C'est une machine équipée de plusieurs processeurs (2, 4 ou 8) avec une performance de 333 MégaFLOPS par processeur et une performance totale record de 2,3 GigaFLOPS.

D'autres moyens pour paralléliser des calculs ont été mis en place et les supercalculateurs se sont rapidement dotés systématiquement de plusieurs processeurs travaillant en parallèle (Hoffman and Traub, 1989). Depuis, la puissance de calcul des supercalculateurs ne cesse de croître. Deux fois par an, depuis juin 1993, une liste des 500 meilleurs supercalculateurs les plus puissants du monde est publiée sur le site http://www.top500.org. Cette liste est considérée comme un classement de référence pour les supercalculateurs. Elle est établie en fonction de la puissance des machines sur la base du test de performance Linpack[3], ce test se trouve maintenant remis en cause pour les architectures et les besoins modernes en calcul. Dans la dernière liste, datant du mois de

[2] FLOPS: nombre d'opérations à virgule flottante par seconde (en anglais "FLoating point Operations Per Second"). C'est une unité de mesure de la vitesse d'un système informatique.

[3] Linpack est un benchmark datant de la fin des années 70 et permettant de mesurer les capacités de calcul en virgule flottante via la résolution d'un système d'équations linéaires.

novembre 2013, la machine chinoise « Tianhé-2 » occupe la première place loin devant « Titan » la plus grosse machine américaine. « Tianhé-2 » a été développée par l'Université des technologies de la défense nationale de Chine, elle possède une performance de plus de 33,86 PétaFLOPS soit 2 fois mieux que « Titan » qui atteint une performance de 17,59 PétaFLOPS. « Tianhé-2 » est dotée de 16 000 nœuds, chacun possédant deux processeurs « Intel Xeon IvyBridge » et trois processeurs « Xeon Phi » pour un total de 3 120 000 cœurs de calcul. En revanche, la consommation électrique de « Tianhé-2 » est énorme avec 17,808 MégaWatt, soit environ 1,9 GigaFLOPS/watt. L'efficacité énergétique de « Tianhé-2 » est à peu près équivalente à celle de « Titan » qui consomme 8,209 MégaWatt, soit environ 2,1 GigaFLOPS/watt, et ce malgré la différence de densité et des technologies d'accélération utilisées.

En ce qui concerne la France, le nouveau supercalculateur « Pangea » de l'entreprise pétrolière « Total » est actuellement le système français le mieux classé. « Pangea » occupe la 14ème position du Top500 mais représente actuellement le plus puissant supercalculateur détenu par une entreprise privée au monde. C'est un supercalculateur conçu par SGI («Silicon Graphics International Corp ») et doté de 110 400 cœurs de calcul pour une performance d'environ 2,1 PétaFLOPS.

Depuis sa création, le Top500 classe les supercalculateurs en se basant sur un benchmark unique qui est nommé Linpack. Cependant, comme nous l'annoncions ci-dessus, ce benchmark compte depuis des années plusieurs détracteurs et subit de plus en plus de critiques. En effet, le test Linpack ne correspond plus aux modèles de calcul utilisés par les applications modernes régies par des équations différentielles. Il n'est plus corrélé aux usages réels des supercalculateurs actuels qui traitent des applications HPC complexes avec un accès aux données selon des modèles irréguliers et ayant tendance à avoir des besoins en efficacité énergétique, en bande passante et en réseaux d'interconnexion à très faible latence aussi importants que la puissance de traitement brute. Ainsi, le Top500 est actuellement considéré par ses détracteurs comme étant un classement déconnecté de la réalité et loin des besoins réels des utilisateurs auxquels il est destiné. Il reste cependant un enjeu politique pour les institutions publics ou privés qui financent ou gèrent les projets (ou les budgets) des supercalculateurs dont certains sont aujourd'hui conçus pour gagner le concours du Top500 et non pas pour réaliser de meilleures performances. Ce débat sur l'utilité de Linpack a poussé J. Dongarra, chercheur à l'université du Tennessee (USA) et auteur de Linpack, à développer une nouvelle suite de

tests nommée HPCG (« High Performance Conjuguate Gradient ») et qui devrait d'abord figurer comme benchmark supplémentaire lors des prochaines versions du Top500, aux côtés de Linpack qui sera remplacé progressivement (Dongarra and Heroux, 2013). HPCG permettra de mieux prendre en considération les modèles de calcul et d'accès aux données couramment trouvés dans les applications modernes, afin de mieux mettre en évidence les différences de performance entre les supercalculateurs actuels.

En outre, de nouveaux classements pour les supercalculateurs ont été proposés, comme le Green500[4] (première liste publiée en novembre 2007), le Graph500[5] (première liste publiée en novembre 2010) et le plus récent Green Graph 500[6] (première liste publiée en juin 2013).

Green500 est une liste des supercalculateurs du Top500 classée selon l'efficacité énergétique calculée en fonction de la performance par watt. L'objectif de Green500 est d'établir un classement des supercalculateurs les plus éconergétiques au monde. En effet, pendant des décennies, la notion de performance des supercalculateurs est synonyme de vitesse, ce qui a conduit à l'émergence des systèmes consommant des quantités extrêmes d'énergie électrique et produisant tellement de chaleur que des installations de refroidissement extravagantes doivent être construites afin d'assurer leur bon fonctionnement.

Le projet Green500 a commencé en Avril 2005 après un discours prononcé par le Dr Wu-Chun Feng lors du workshop IEEE « High-Performance, Power-Aware Computing », au Colorado (USA). Il a ensuite été formellement proposé un an plus tard (Sharma et al., 2006). La première liste Green500 a été publiée en novembre 2007. Depuis, deux mises à jour de la liste Green500 sont publiées annuellement. Selon la dernière version (novembre 2013), la machine japonaise « TSUBAME-KFC » du centre « GSIC » de l'Institut de technologie de Tokyo occupe la tête du classement des superordinateurs les plus efficaces au monde en consommation d'énergie, avec 4,5 GigaFLOPS/watt. Elle devance « Wilkes » de l'Université de Cambridge, la machine européenne la plus efficace à 3,6 GigaFLOPS/watt. Une deuxième machine japonaise « HA-PACS » du « Center for Computational Sciences » de l'université de Tsukuba complète le podium avec 3,5

[4] http://www.green500.org/

[5] http://www.graph500.org/

[6] http://green.graph500.org/

GigaFLOPS/watt. Les deux machines « Tianhé-2 » et « Titan » qui occupent les deux premières places du Top500 sont respectivement classées 40ème et 35ème du Green500.

2.4. « Symmetric MultiProcessor »

Un multiprocesseur symétrique (SMP pour « Symmetric MultiProcessor ») est une machine parallèle à mémoire partagée (de type MIMD-SM). Chaque processeur d'une machine SMP peut exécuter une tâche indépendamment des autres processeurs mais peut accéder à toutes les ressources de la machine. Ces ressources ainsi que l'ensemble des processeurs sont gérés par un seul système d'exploitation. Cette architecture consiste à augmenter le nombre de processeurs d'une machine tout en utilisant une même mémoire. L'accès se fait par un bus système commun (figure 19) connecté aux entrées/sorties (I/O) du système.

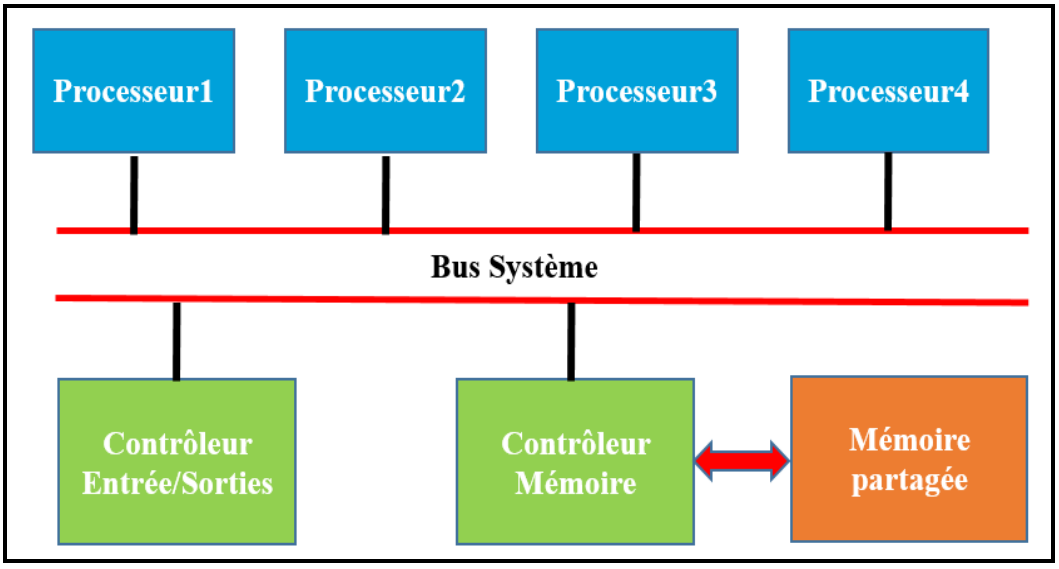

Figure 19. Architecture matérielle de base pour un SMP: exemple de machine SMP avec quatre processeurs (4 sockets sur une carte mère par exemple).

L'avantage de cette architecture est sa capacité d'extension à un coût raisonnable: des processeurs supplémentaires peuvent être rajoutés à la machine pour augmenter sa puissance de calcul. Cependant, cette capacité d'extension a une limite. En effet, le nombre maximum de processeurs dans une machine SMP reste limité par les conflits d'accès au niveau du bus système et du système d'exploitation. Lorsque plusieurs processeurs sont ajoutés, les performances du SMP sont éventuellement limitées par la saturation du bus système partagé. De plus, pour une seule machine, le système d'exploitation a ses limites

lorsqu'on travaille sur des applications gourmandes en temps de calcul, surtout avec la gestion des conflits d'accès aux ressources. Cette limite a été très sensiblement franchie par Silicon Graphics avec des nœuds d'interconnexion propriétaires très innovants. Fin 2013, les limites pour des machines Silicon Graphics de type UV sont de l'ordre de 2048 cœurs de calcul et pour les autres machines le maximum est atteint avec 80 cœurs sur des processeurs Intel particuliers dédiés aux serveurs et possédants 10 cœurs de calcul.

Pour accéder une puissance de calcul plus importante, une alternative à cette architecture est celle du cluster de calcul (architecture MIMD-DM).

2.5. Ferme de calcul ou « Computing Cluster »

Appelé aussi « grappes de serveurs » ou encore de « multi-ordinateurs », un cluster de calcul est un ensemble de machines indépendantes regroupées physiquement en un même lieu. Ces machines sont appelées nœuds, chaque nœud peut-être une machine SMP (ce qui est le cas aujourd'hui). La somme des puissances de calcul élémentaires de tous les nœuds forme la puissance du cluster pour fournir ainsi une plus grande capacité de calcul et de stockage distribuée. Cette capacité est techniquement très facile à étendre. Les différents nœuds du cluster communiquent à travers un réseau local à haut-débit pour garantir le transfert rapide des données. Le coût et la performance du cluster dépend fortement de la qualité et de la performance du réseau d'interconnexion. Lorsqu'un ou plusieurs nœuds tombent en panne, le cluster continue à fonctionner en utilisant les nœuds restants.

Il existe deux principaux types de cluster (Trelles, 2000):

- ✓ **COW (« Cluster of Workstations »)**: cette architecture consiste à regrouper des ordinateurs hétérogènes connectés par un réseau haut-débit.

- ✓ **Beowulf**: le nom Beowulf a été originalement introduit en 1994, par Thomas Sterling et Donald Becker de la NASA comme référence à un ordinateur parallèle pour le calcul scientifique qu'ils ont développé (Sterling et al., 1995). Un cluster Beowulf est un ensemble homogène de machines regroupées et connectées par un réseau local (www.beowulf.org). Il est souvent constitué de plusieurs nœuds clients et un seul nœud serveur (« Master ») qui contrôle tout le cluster (les grands clusters Beowulf peuvent avoir plus qu'un nœud « Master » et/ou d'autres nœuds pour gérer des tâches particulières telles que la communication avec l'extérieur). C'est

l'architecture la plus répondue des clusters. La figure 20 représente un exemple simple de cluster Beowufl contenant quatre nœuds dont un maître (« Master ») et trois esclaves (« Slaves »).

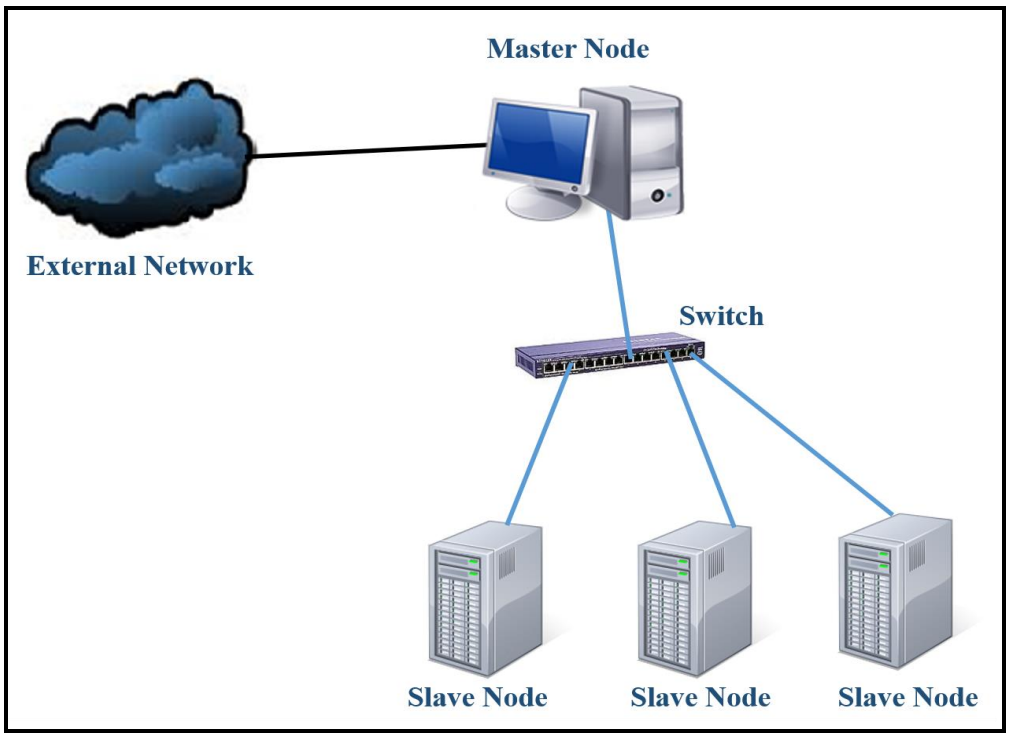

Figure 20. Schéma général d'un cluster de type Beowulf.

Les clusters ont été largement utilisés pour exécuter des applications bioinformatiques nécessitant un temps de calcul important. Dans le cas des biopuces, plusieurs outils de design de sondes s'exécutent sur un cluster, tels que PhylArray (Militon et al., 2007), HiSpOD (Dugat-Bony et al., 2011) et KASpOD (Parisot et al., 2012). Des logiciels de recherches de similarité, tels que Blast (version MPI-Blast) ou d'alignement de séquences tel que ClustalW (version ClustalW-MPI) (Li, 2003) ont été aussi implémentés sur un cluster. Cependant, avec la croissance continue des données génomiques et l'explosion des quantités de séquences des banques de données, les clusters ne sont pas toujours suffisants pour faire face à des besoins de calcul massifs.

2.6. Grille de calcul

L'architecture de grille de calcul est une approche distribuée qui consiste à relier et partager des ressources hétérogènes de calcul et/ou de stockage de plusieurs centres de calcul distants, pour passer d'une puissance de calcul de quelques centaines de processeurs

pour un cluster à une capacité de calcul et de stockage quasi illimitée dans le cadre d'une grille de calcul (plusieurs centaines de milliers de cœurs de calcul et plus d'une centaine de Petaoctets de données). A part sa grande capacité en ressources informatiques, un autre aspect important de la grille de calcul est son interopérabilité. En effet, une grille doit être capable de fonctionner avec différents produits ou systèmes informatiques, sans restriction d'accès ou de mise en œuvre.

La notion de grille a été inspirée du réseau de distribution électrique dont les utilisateurs consomment de l'électricité sans se préoccuper de sa source ou de la complexité du réseau. En 1999, le terme « grille » est pour la première fois formalisé et défini par Ian Foster et Carl Kesselman (Foster and Kesselman, 1999): « *A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities* ». Cet ouvrage d'Ian Foster et Carl Kasselman a introduit les notions de base des grilles et est devenu par la suite une référence dans ce domaine. Depuis la naissance de ce nouveau concept, l'intérêt porté aux grilles de calcul ne cesse de croître et les années 2000 ont été marquées par une véritable émergence des grilles de calcul. En effet, plusieurs institutions, inspirées par le succès de ce concept, ont décidé de développer leurs propres grilles telles que: « Information Power Grid IPG » de la NASA, « Alliance Grid » (Ferguson and Towns, 2001), « National Partnership for Advanced Computational Infrastructure NPACI Grid »[7], « DOE Science Grid »[8], « EU DataGrid »[9], « TeraGrid »[10] et la grille européenne EGI (« European Grid Infrastructure »)[11] qui a succédé à l'ancien projet EGEE (« Enabling Grid for E-sciencE ») qui a pris fin en 2010 et qui était initié par DataGrid. L'EGI est la grille de calcul que nous utilisons dans le cadre de cette thèse. L'EGI est une infrastructure européenne distribuée, à financement public. C'est le fruit d'une collaboration entre des initiatives nationales de grille et des laboratoires de recherche européens, dont l'objectif est de fournir à la communauté scientifique européenne un accès à de grandes ressources de calcul et de stockage à travers les techniques de grilles de calcul, pour soutenir la recherche

[7] npacigrid.npaci.edu

[8] www.doesciencegrid.org

[9] eu-datagrid.web.cern.ch/eu-datagrid

[10] www.teragrid.org

[11] http://www.egi.eu/

et l'innovation en Europe. L'EGI offre actuellement plus de 370 000 cœurs de calcul et 170 pétaoctet de capacité disque, avec des ressources distribuées sur environ 350 centres répartis sur 56 pays en Europe, Asie-Pacifique, Canada et Amérique latine.

En 2006, un organisme appelé « Open Grid Forum » OGF a été créé pour la standardisation des grilles de calcul. L'OGF est une communauté ouverte d'utilisateurs, développeurs et fournisseurs, ayant pour objectif d'appuyer les efforts de standardisation pour le calcul distribué et de soutenir le développement et le partage de standards et spécifications pour les grilles de calcul. Elle promeut également le calcul sur grille comme solution pour le développement d'applications dans les entreprises et les établissements de recherche universitaires. L'OGF est composé de milliers de membres issus de plus de 400 organisations en industrie et recherche scientifique dans plus de 50 pays. Plusieurs standards ont été générés par l'OGF: DFDL (« Data Format Description Language »), SAGA (« Simple API for Grid Application »), GridRPC (« Grid Remote Procedure Call »), GridFTP (« Grid File Transfer Protocol »), etc.

En France, des initiatives ont été également initiées. Dans ce cadre, une structure CNRS appelée « Institut des Grilles » a été créée en 2007 pour assurer la coordination de l'ensemble des travaux du CNRS utilisant les grilles et le « Cloud Computing ». Cette structure est devenue en 2011 « l'Institut Des Grilles et du Cloud »[12]. Elle gère aussi le « Groupement d'Intérêt Scientifique France Grilles »[13] qui a été créé en 2010 par huit établissements scientifiques pour coordonner le déploiement et la gestion des ressources grilles au niveau national pour la communauté scientifique. A part la participation des centres de calcul français dans le projet de la grille de calcul européenne EGI, d'autres initiatives nationales existent telle que l'infrastructure RENABI GRISBI (« Grid Support for Bioinformatics »)[14] qui est le fruit de la collaboration entre six plateformes nationales de bioinformatique (PRABI Lyon, GenOuest Rennes et Roscoff, CBiB Bordeaux, BIPS Strasbourg, CIB Lille, MIGALE Jouy-en-Josas) dans le cadre du réseau RENABI établi en 2004. L'objectif de cette infrastructure est de permettre le développement et l'exécution d'applications biologiques traitant de grandes quantités de données et de partager les

[12] http://www.idgrilles.fr/

[13] http://www.france-grilles.fr/

[14] http://www.grisbio.fr/

ressources nationales (matériel et logiciel) dédiées à la bioinformatique, à l'aide de composants logiciels de type grille de calcul.

2.6.1. Middleware

La mise en place d'une grille de calcul nécessite l'utilisation d'un middleware (intergiciel) qui s'interface entre les ressources et les applications pour faire collaborer tous les éléments de la grille. Le middleware assure la gestion des jobs (soumission, ordonnancement), des données (transfert, stockage, réplication), de la sécurité et des utilisateurs. Plusieurs middlewares existent. On peut citer « Globus Toolkit », « Glite », « EMI Grid Middleware », « Advanced Resource Connector » ARC, « UNICORE », « iRODS », etc. Le middleware le plus populaire est « Globus Toolkit » (Foster and Kesselman, 1997; Foster, 2001, 2006). C'est un logiciel open source qui contient une suite d'outils pour la sécurité, la gestion des ressources et des données, la gestion des exécutions, la communication, l'accès aux informations, la détection des erreurs et la portabilité. Ces outils peuvent être utilisés ensemble ou séparément pour développer des applications sur grilles de calcul. La figure 21 présente les différents composants de l'architecture du « Globus Toolkit » dans sa dernière version GT5.

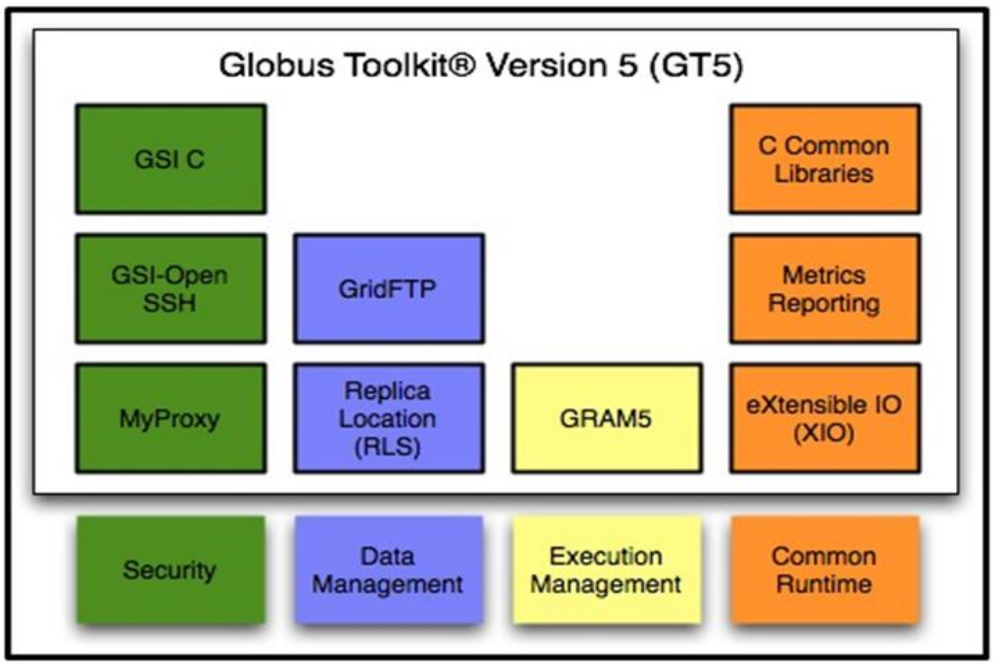

Figure 21. Architecture de la version GT5 du « Globus Toolkit » (Source http://toolkit.globus.org/toolkit/about.html, novembre 2013).

D'un autre côté, une collaboration entre les quatre principaux fournisseurs européens de middleware pour le calcul distribué, à savoir ARC, dCache, gLITE et UNICORE, a donné naissance à l'EMI (« The European Middleware Initiative »). Il s'agit d'une plateforme logicielle pour le calcul intensif distribué. Le projet EMI a démarré en mai 2010 avec le soutien de la commission européenne et de plusieurs instituts de recherche en Europe, afin de développer un ensemble de composants d'intergiciels fiable, cohérent, stable et commun pour toutes les infrastructures informatiques distribuées en Europe. Il permet d'introduire, en Europe et ailleurs, de nouvelles fonctionnalités pour répondre aux besoins des utilisateurs en matière de sécurité, calcul, données et infrastructure. Les solutions de l'EMI sont utilisées par les communautés de recherche scientifique et différentes infrastructures grilles telles que EU-IndiaGrid, WLCG « Worldwide LHC Computing Grid » et EGI (« Grid Infrastructure »). Pour la collaboration avec l'EGI, la solution de middleware EMI3 fait actuellement partie de l'UMD (« Unified Middleware Distribution ») utilisé par l'EGI. En effet, l'EGI ne développe pas son propre middleware, mais utilise des composants développés par des fournisseurs de cette technologie et les intègre dans l'UMD. Ce dernier peut être donc défini comme étant l'ensemble de composants logiciels développés pour l'EGI par des fournisseurs externes et déployés sur l'infrastructure grille pour assurer son bon fonctionnement. Afin de garantir un fonctionnement fiable et de haute qualité de l'EGI, tous les composants de l'UMD doivent passer par un processus d'entretien et de vérification qui consiste à définir et vérifier les exigences fonctionnelles, de performance et de qualité, et à réaliser l'intégration, le déploiement et les tests des logiciels dans les environnements de production.

2.6.2. Fonctionnement des grilles de calcul

Le fonctionnement d'une grille de calcul repose sur la notion d'organisations virtuelles (VO pour « Virtual Organizations ») (Foster et al., 2001) qui désigne des communautés d'utilisateurs qui partagent un intérêt ou un domaine d'utilisation commun sur la grille. Par exemple, on peut citer la VO « Biomed » qui regroupe les utilisateurs dans le domaine médical et de biologie. L'intérêt d'une VO est d'organiser, de simplifier et de sécuriser l'accès aux données partagées par l'organisation, ainsi que définir les droits ou les restrictions d'accès aux différentes ressources de la grille.

Pour pouvoir utiliser les ressources de la grille pour exécuter ses applications, l'utilisateur doit se servir d'une interface utilisateur (UI pou « User Interface »). Il s'agit d'une machine de connexion à la grille sur lequel l'utilisateur doit posséder un compte et

installer son certificat grille (clé publique, clé privée). Ce certificat sera nécessaire pour que l'utilisateur puisse créer un proxy VOMS (« VO Membership System »), une étape obligatoire avant de soumettre ses jobs sur grille. En cas de besoin, l'utilisateur peut stocker les fichiers nécessaires pour son application ou les résultats de l'exécution de ces jobs sur la grille à l'aide de serveurs de stockage (SE pour « Storage Element ») et le catalogue de fichiers (LFC pour « LCG File Catalog »). Les SEs fournissent l'espace disque nécessaire pour le stockage des données tandis que le LFC fournit une association entre l'identifiant et les localisations des fichiers sur la grille pour permettre à l'utilisateur d'accéder à ses fichiers sans savoir nécessairement le SE sur lequel ils sont stockés. L'utilisateur peut ensuite soumettre ses jobs à la grille. Ces jobs seront gérés et ordonnancés à l'aide du système de gestion des jobs (WMS pour « Workload Management Server ») et du Système d'information de la grille (BDII pour « Berkeley Database Information Index »). Les jobs seront transférés aux éléments de calcul (CE pour « Computing Element ») correspondant désignés lors de l'ordonnancement. Un CE représente l'ensemble des ressources de calcul présentes sur un site et est composé de plusieurs nœuds de calcul (WNs pour « Worker Node ») sur lesquels les jobs seront exécutés. Le temps d'attente des jobs peut largement varier en fonction de la disponibilité des CEs auxquels ils sont affectés.

La figure 22 illustre les composants et les étapes du fonctionnement général d'une grille.

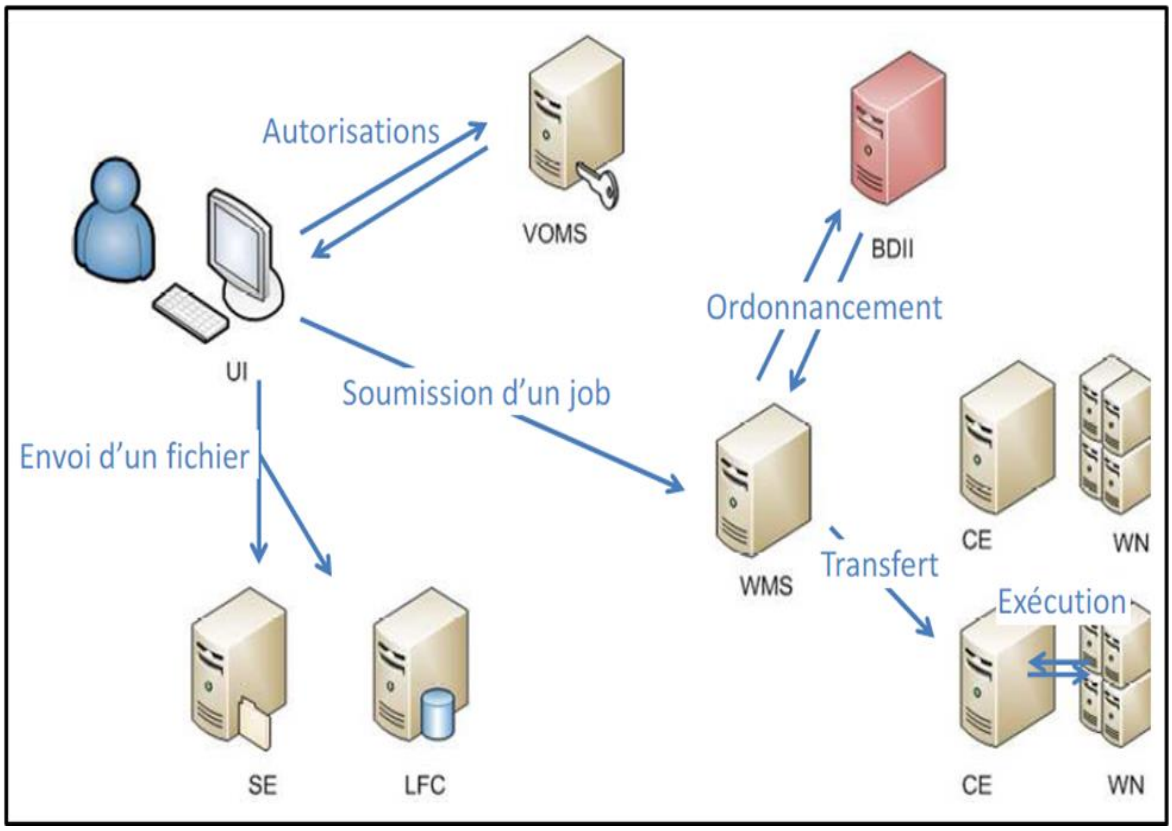

Figure 22. Fonctionnement général de la grille.

2.6.3. L'utilisation de la grille pour les applications de biopuces

Depuis leur naissance, les grilles de calcul ont apporté des solutions concrètes aux problématiques à haut-débit des Sciences de la Vie (Konagaya, 2006), afin de faire face à la puissance de calcul considérable nécessaire pour certaines applications biologiques telles que l'annotation des génomes, la prédiction de fonction de protéines, le criblage (« docking ») moléculaire et l'interaction spatio-temporelle des molécules, etc. Les applications de biopuces à ADN entrent aussi dans le cadre des applications complexes nécessitant une puissance de calcul et de stockage très importante (Maglogiannis et al., 2007). Dans ce cadre, des applications de type grille de calcul, en relation avec la conception et le traitement des données de biopuces, ont été développées. Par exemple, GEMMA (« Grid Environment for Microarray Management and Analysis ») (Beltrame et al., 2007) est un portail web créé pour stocker et partager des expériences de biopuces ainsi que des outils standards et fiables pour la gestion et l'analyse des données. GRISSOM (« GRids for In Silico Systems BiOlogy and Medicine ») (Chatziioannou et al., 2011) est une solution Web exploitant la puissance des grilles de calcul pour la gestion, la recherche et la diffusion des connaissances biologiques dans le contexte de l'analyse des profils

d'expression des gènes à l'échelle du génome. GRISSOM permet l'interprétation et le stockage des expériences de biopuces à ADN. D'autres applications de type grilles pour la gestion et l'analyse des données issues des expériences de biopuces existent: XPS (« eXpression Profiling System ») (Stratowa, 2003), "Grid Portal for microarrays » (Porro et al., 2007), HECTOR (Kanaris et al., 2009), etc.

En ce qui concerne la sélection de sondes pour biopuces à ADN en utilisant les grilles de calcul, une initiative de distribution de la sélection de sondes pour biopuces phylogénétiques sur la grille de calcul européenne a été présentée dans (Missaoui, 2010). Les tests de spécificité qui consistent à chercher les similarités de séquences, représentent l'étape la plus consommatrice de ressources de calcul dans un outil de sélection de sondes. Cette étape est généralement réalisée par le fameux programme Blast qui a une nature extrêmement parallèle facilitant sa distribution sur des plates-formes de type grille de calcul (Afgan et al., 2006). Dans ce contexte, plusieurs versions de Blast distribuées sur grilles de calcul ont été proposées ces dernières années: Konishi et al., 2003; Kumar et al., 2004; Bayer et al., 2005; Krishnan et al., 2005; Dowd et al., 2005; Carvalho et al., 2005; Afgan et al., 2006; Trombetti et al., 2007; Sun et al., 2007; Mirto et al., 2008; Missaoui, 2009, etc.

2.7. Informatique en nuage ou « Cloud computing »

Le principe du cloud peut être résumé comme la mise à disposition du client, à sa demande et via le web, d'un ensemble partagé de ressources informatiques (applications, réseaux, matériel pour le calcul et/ou le stockage). Ces ressources sont accessible via le web sous forme de services virtuels payants, extensibles, sécurisés et adaptables en fonction des besoins du client. Le prestataire de ces services doit assurer la disponibilité et la sécurité optimale des ressources louées dont l'emplacement et le fonctionnement ne sont pas connus par les clients. Le cloud se caractérise par une grande flexibilité et une virtualisation des ressources.

Le cloud peut être classé en quatre différents types:

- ✓ **Cloud public**: comme son nom l'indique, c'est un cloud ouvert au grand public (internaute au sens large) offrant des ressources partagées à la demande. Les ressources d'un cloud public sont gérées par un fournisseur tiers et sont partagées, via Internet, par plusieurs entités (personnes ou entreprises).

- ✓ **Cloud privé**: c'est un cloud à accès restreint, exploité exclusivement par une seule entité. Les ressources sont soit géographiquement situées dans les locaux de l'entreprise (cloud privé interne) ou chez le fournisseur (cloud privé hébergé ou externe). Un cloud privé externe n'est accessible que par les utilisateurs autorisés, via des réseaux sécurisés VPN. La solution du cloud privé est adaptée aux grandes entreprises ou à celles dont la criticité et la sécurité des données sont très importantes.

- ✓ **Cloud communautaire**: c'est un cloud utilisé exclusivement par une communauté spécifique d'organisations qui ont des besoins ou intérêts communs. Il peut être détenu et géré sur les locaux par un ou plusieurs membres de cette communauté, ou en dehors des locaux par un fournisseur de l'extérieur.

- ✓ **Cloud hybride**: c'est un environnement cloud mixte dont le client utilise des ressources de deux ou plusieurs clouds de types différents (privés, publics ou communautaires) mais liés par une technologie permettant la disponibilité des données et la portabilité des applications. Une organisation qui exploite un cloud hybride peut par exemple utiliser le cloud public seulement lors de pics d'activité et utiliser le cloud privé le reste du temps. Elle peut aussi utiliser un cloud pour les données et un autre pour les applications. Le cloud hybride offre une alternative intéressante qui combine les avantages des clouds privés et publics, mais sa mise en œuvre nécessite cependant une étude personnalisée en fonction des besoins du client. Selon une étude réalisée par IBM en octobre 2013 auprès de 802 entreprises et décisionnaires cloud (Comfort et al., 2013), 44% des entreprises interrogés et ayant déployé un cloud à grande échelle et réalisé des gains de compétitivité, utilisent le cloud hybride.

On peut également distinguer trois modèles d'utilisation du cloud:

- ✓ **SaaS « Software as a Service »**: Le principe est de mettre à la disposition du client, en fonction de ses besoins, des applications accessibles à travers le web. C'est le fournisseur cloud qui contrôle et maintient les applications, les bases de données, les serveurs, le stockage, les réseaux et tous les autres services (Mell and Grance, 2011). Le client ne gère aucun composant ou service. Dans ce modèle, le client n'achète plus de licences pour acquérir une application, mais il utilise les

applications sur cloud à la demande et paye en fonction de son utilisation (durée et nombre d'utilisateurs par exemple). Cette approche se distingue par la facilité d'utilisation de ses services (pas de déploiement logiciel, interface web) et son accessibilité (téléphone par exemple). Cependant, des problèmes au niveau de la sécurité (accès et confidentialité des données) et de la pérennité des services peuvent exister (Blanchet and Loomis, 2010). Comme exemples pour ce modèle, on peut citer les services Google Apps[15] et celles de SalesForce[16].

- ✓ **PaaS « Platform as a service »**: Le client gère uniquement l'application et son déploiement, tandis que le fournisseur cloud maintient le reste: les bases de données, les serveurs, le stockage, les réseaux, et les autres services. Les applications doivent être développées avec des outils (langages de programmation, librairies, etc.) supportés par la plateforme. Ces applications ne sont pas alors forcément compatibles avec d'autres plateformes. Dans le cadre de ce modèle, on trouve des plateformes comme Google App Engine[17] et Windows Azure[18].

- ✓ **IaaS « Infrastructure as a Service »**: Le principe est de fournir au client une machine virtuelle avec les ressources demandées. Le client a alors l'impression d'utiliser une véritable machine physique. Dans ce modèle, le fournisseur maintient la virtualisation, le matériel serveur, le stockage, les réseaux. Le client maintient les applications, les bases de données et le système d'exploitation. L'avantage de ce modèle est que l'utilisateur possède le contrôle total de sa ressource virtuelle. Une telle approche permet également au client d'exécuter des environnements personnalisés et accessibles à tout moment (Blanchet and Loomis, 2010). Parmi les

[15] https://cloud.google.com/products/app-engine/

[16] http://www.salesforce.com/fr/

[17] https://cloud.google.com/products/

[18] www.windowsazure.com/

IaaS, on trouve par exemple Amazon Web Services[19], GoGrid[20], FlexiScale[21], ElasticHosts[22].

Malgré les questions et les inquiétudes des clients potentiels du cloud (niveau de sécurité des données, disponibilité et continuité des données et services, compatibilité et portabilité des applications, coûts, etc.), le cloud possède de nombreux avantages:

- ✓ Le client possède les dernières versions et mise à jours des logiciels sans besoin d'une assistance informatique locale.
- ✓ Le client peut réduire les coûts via la location de logiciels en nuages puisque il paie seulement ce qu'il consomme.
- ✓ Le client possède des services (matériel, logiciel et stockage) modulable et extensible sur demande.
- ✓ Le client évite tous les problèmes de maintenance et de gestion des ressources informatiques.
- ✓ C'est le fournisseur qui s'occupe des problèmes de sécurité des données.
- ✓ Les données sont accessibles via le web à partir de n'importe quel endroit.

2.8. Calcul hybride

La réduction de l'augmentation de la puissance des microprocesseurs, leur limite au niveau fréquence et aussi en calcul flottant, a poussé la communauté du calcul intensif à avoir recours à une autre forme de processeurs: les processeurs graphiques (GPU pour « Graphical Processing Unit »). C'est à la fin des années 1990 et le début des années 2000, que les premiers travaux qui exploitent la puissance des processeurs graphiques pour effectuer du calcul flottant généraliste sont apparus (Rumpf and Strzodka, 2001; Harris et al., 2002; Govindaraju et al., 2004, etc.). Cet intérêt pour les GPGPUs (« General-Purpose computation on Graphics Processing Units » - terme utilisé pour désigner les GPUs utilisés

[19] http://aws.amazon.com/fr/

[20] http://www.gogrid.com/

[21] http://www.flexiscale.com/

[22] http://www.elastichosts.com/

à des fins de calculs généralistes) a été encouragé par l'apparition d'API permettant de programmer des instructions classiques sur GPU: OpenGL (Woo et al., 1999), DirectX (Bargen and Donnelly, 1998). Cependant, le lancement, en 2007, de la nouvelle API CUDA[23] (« Compute Unified Device Architecture ») pour coder et exécuter des programmes généralistes écrits par exemple en C sur des processeurs graphiques NVIDIA surpuissants, a véritablement marqué l'histoire des GPGPUs et a participé à leur diffusion comme solution de calcul intensif. En 2008, un standard ouvert combinant une API et un langage de programmation dérivé du C et permettant de coder des applications généralistes sur carte graphique, a été proposé par le « Khronos Group »: OpenCL[24] (« Open Computing Language »).

Bien que les GPGPUs représentent une solution de calcul efficace pour améliorer les performances de certains programmes, l'apport de cette architecture pour d'autres programmes reste négligeable ou même négatif. C'est le cas par exemple des programmes manipulant de grandes quantités de données. Comme alternative, des machines hybrides possédant à la fois des processeurs très puissants et des cartes GPGPU ont été utilisées pour permettre ainsi aux applications d'exploiter le meilleur outil entre les CPUs et les GPUs. Aujourd'hui on trouve ce type de machines hybrides parmi les supercalculateurs, mais on les trouve également sous forme de stations de travail à coût réduit, notamment depuis l'introduction du « personnal supercomputer » à base de carte Tesla fin 2008 à la Supercomputing Conference d'Austin (Texas).Pour plus de détails sur le calcul hybride, le lecteur pourra se référer à (Caux, 2012).

Les processeurs graphiques ont été employés, depuis 2007, pour accélérer les applications bioinformatiques en liaison avec les biopuces à ADN, comme l'alignement de séquences et la recherche de similarité (Schatz et al., 2007; Manavski and Valle, 2008; Ling and Benkrid, 2010; Vouzis and Sahinidis, 2011; Liu et al., 2011; Blazewicz et al., 2013), la prédiction des structures secondaires (Rizk and Lavenie, 2009) et l'analyse des données issues de biopuces (Carpenter, 2009; Shterev et al., 2010; Buckner et al., 2010).

[23] http://www.nvidia.com/object/cuda_home_new.html

[24] http://www.khronos.org/opencl

2.9. Comment utiliser le calcul intensif dans les développements informatiques pour les biopuces exploratoires

Les besoins en temps de calcul d'un programme de sélection de sondes exploratoires pour l'étude des environnements complexes sont en augmentation continue (jusqu'à plusieurs jours de calcul pour la sélection de sondes pour un seul groupe de séquences) tout comme le nombre de séquences déposées dans les bases de données génomiques. Un outil de sélection de sondes doit donc pouvoir traiter une grande masse de données à l'entrée et à la sortie du programme, en un temps raisonnable. Le calcul intensif représente une solution permettant d'augmenter les performances et diminuer la complexité de tels programmes complexes. Cependant, les besoins de calcul de ces programmes sont souvent fortement dépendants de la base de données utilisée ainsi que de la densité, du type et de l'application de biopuces. Dans ce cadre, grâce à la variété des architectures de calcul intensif et à leurs différents niveaux de puissance, nous avons décidé d'utiliser différentes infrastructures à savoir SMP, cluster et grille de calcul, pour pouvoir proposer la solution la mieux adaptée en fonction des besoins de l'utilisateur. Cependant pour réaliser une telle application générique, sans pour autant augmenter sa complexité, l'utilisation d'une approche d'Ingénierie Dirigée par les Modèles nous a paru la plus adaptée pour un développement moderne avec transformation/génération de code.

3. L'ingénierie dirigée par les modèles

L'ingénierie dirigée par les modèles (IDM) est une approche issue du génie logiciel et qui a marqué une évolution radicale dans la manière de développer des applications informatiques. L'IDM donne un cadre méthodologique et technologique pour unifier plusieurs approches dans un processus global homogène, tout en permettant de sélectionner la technologie la mieux adaptée à chaque étape du développement (Favre et al., 2006). Selon Favre et al. (2006), « *pour faire face à la complexité et à l'évolution croissante des applications, l'IDM ouvre de nouvelles voies d'investigation. Cette approche vise non seulement à favoriser un génie logiciel plus proche des métiers […] mais elle intègre également comme fondamentales la composition et mise en cohérence de ces perspectives […]. L'IDM cible une production logicielle bien fondée […]* ».

L'IDM utilise intensivement et systématiquement les modèles dans tout le processus du développement logiciel. Les modèles deviennent ainsi l'outil central dans le développement, le fonctionnement et l'évolution des systèmes informatiques complexes.

En effet, avec l'IDM, les notions code et objet ne sont plus le cœur du développement logiciel, ce sont d'autres notions qui prennent cette place centrale: le modèle, le méta-modèle et la transformation de modèles (Greenfield and Short, 2003) (figure 23). Avec l'IDM, un système peut être décrit par plusieurs modèles liés les uns aux autres. Par exemple, le code de l'application est obtenu par transformation d'un modèle métier plus abstrait en un autre plus concret représentant le système selon une technologie donnée. En effet, l'IDM est une approche intégrative dans laquelle tout ou partie du code de l'application informatique peut être générée à partir de modèles.

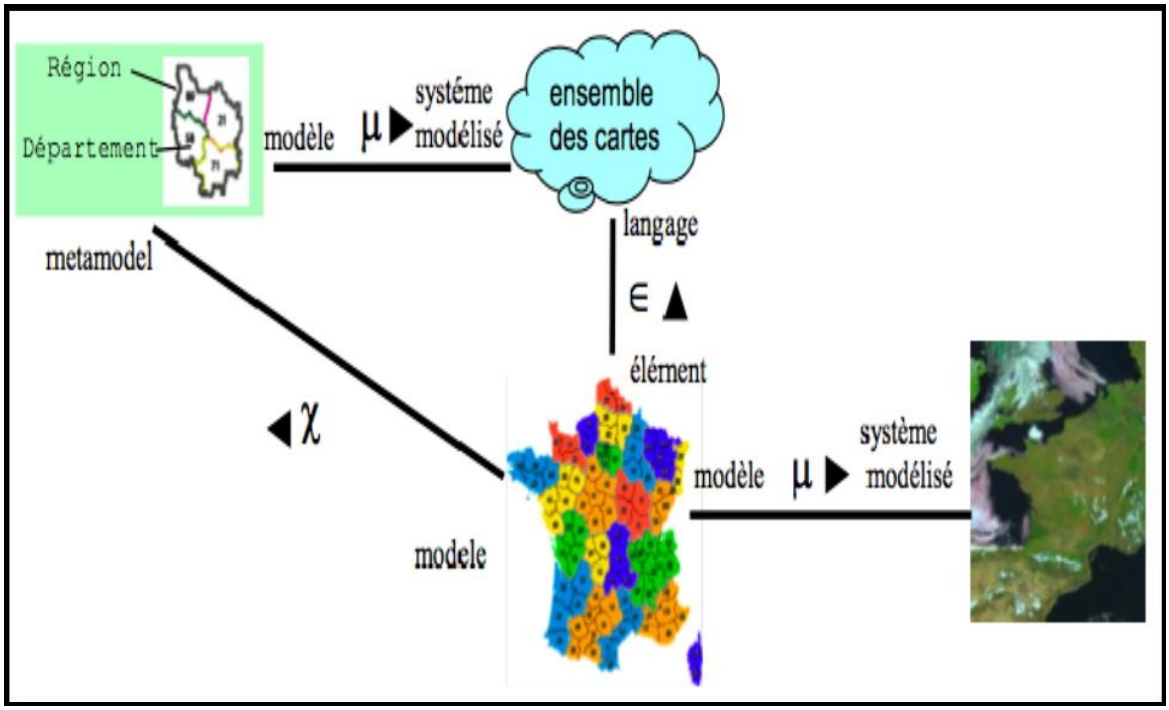

Figure 23. Relations entre système, modèle, métamodèle et langage suivant un exemple de carte administrative tiré de (Favre et al., 2006).

L'IDM est une approche unificatrice et générale qui peut être considérée comme une famille d'approches (Bézivin, 2005). Dans ce cadre, plusieurs approches ou variantes mettant en œuvre les principes de l'IDM existent. Ces approches partagent les concepts mais pas forcément les standards. Parmi ces approches on trouve « Software Factories » de Microsoft (GreenField and Short, 2003), le MIC (« Model Integrated Computing ») (Sztipanovits and Karsai, 1997), MDD (« Model Driven Development ») (Cetinkaya et al., 2011), MDA (« Model Driven Architecture »)[25], etc.

[25] http://www.omg.org/mda/

3.1. « Model Driven Architecture » MDA

La première variante de l'IDM est l'architecture dirigée par les modèles (MDA pour « Model Driven Architecture ») (figure 24). Elle a été proposée au début des années 2000 par l'OMG (« Object Management Group »). La MDA est en fait une collection de standards industriels proposée par l'OMG en novembre 2000, cette collection vise à promulguer de bonnes pratiques de modélisation grâce à une exploitation avantageuse des modèles (Combemale, 2008). Pour assurer le développement et la maintenance des applications logicielles avec une méthode dirigée par les modèles qui sépare les contraintes fonctionnelles des contraintes techniques, l'approche MDA se base sur la séparation de la description de la partie des systèmes indépendants des plateformes (PIM pour « Platform Independent Model ») de celle spécifique à une plateforme (PSM pour « Platform Specific Model ») (Favre et al., 2006). Cette séparation des spécifications fonctionnelles et techniques ou des Pim et SIM permet de garantir la réutilisation des PIM, leur portabilité et l'interopérabilité. Par exemple, en cas de changement d'architecture technique, il suffit de générer, à partir du même PIM, un nouveau PSM.

La première étape dans une démarche MDA consiste à construire un PIM qui représente le modèle métier, à l'aide des technologies UML, MOF (« Meta-Object Facility ») et CWM (« Common Warehouse Meta-Model »). Le PIM construit est ensuite transformé en PSM représentant un modèle des parties techniques de la plateforme cible basée sur des technologies telles que EJB (Entreprise Java Beans), CORBA (Common Object Request Broker Architecture), .NET, XMI (XML Metadata Interchange) ou les web services. Cette transformation permettra l'implémentation concrète du système. Enfin, un code applicatif est généré à partir du PSM, pour obtenir une implémentation exécutable pour une plate-forme spécifique.

Plusieurs outils ont été proposés pour effectuer la transformation des PIM en PSM suivant les standards MDA. Comme exemple, on trouve: OptimalJ, ACCELEO[2], ATLAS Transformation Language[3], MODEL-IN-ACTION, Mia-Studio, ANDROMDA[5], OpenMDX[6], MDE Eclipse[7], Blueprint ME[8], Visual Studio, etc.

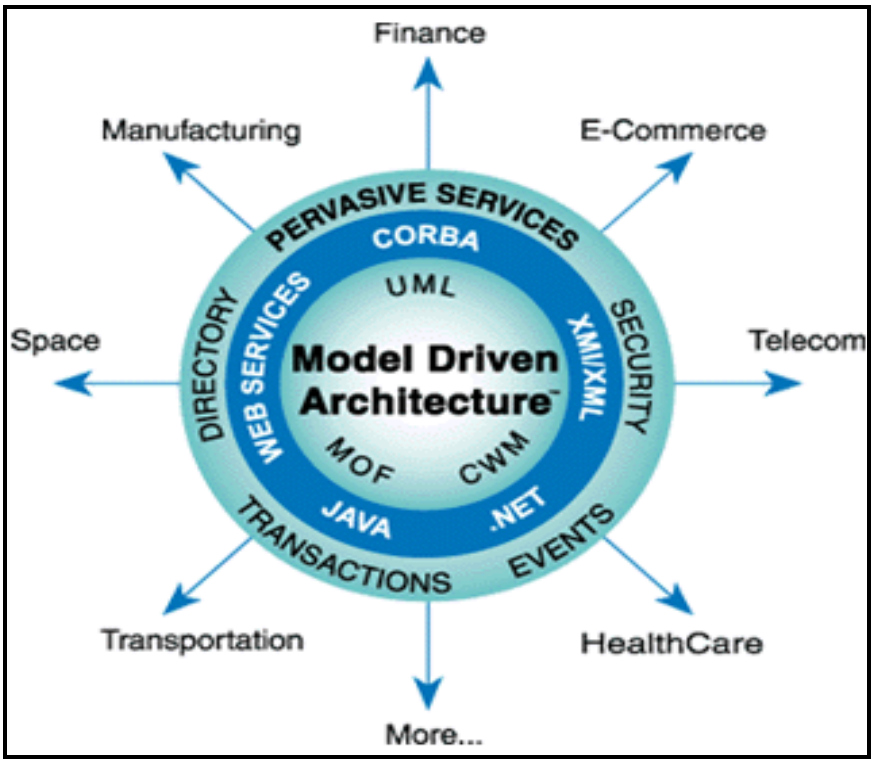

Figure 24. L'architecture MDA (Source http://www.omg.org/mda/, avril 2014).

3.2. Concepts fondamentaux de l'IDM

Avant la naissance de l'IDM, le principe fondamental sur lequel se basaient la plupart des approches de génie logiciel à partir des années 80 était « tout est objet » (« Everything is an object »). Avec l'IDM, le système n'est plus un objet du point de vue abstraction, mais plutôt un modèle; un nouveau principe apparait alors: « tout est modèle » (« Everything is a model ») (Bézivin, 2005). Ainsi, l'IDM s'appuie sur trois concepts fondamentaux:

- ✓ les modèles;
- ✓ les métamodèles;
- ✓ la transformation de modèles.

3.2.1. Les modèles et la relation « représentation de »

Un modèle peut être considéré comme une représentation simplifiée du système réel étudié (Atkinson and Kuhne, 2003; Seidewitz, 2003; Bézivin, 2004). En effet, le modèle n'est pas censé contenir toutes les informations sur le système réel, mais il doit tout simplement être suffisamment bon pour comprendre le système modélisé et répondre aux

questions qu'on peut poser sur lui dans le cadre d'un objectif précis (Favre, 2004b). Cela implique une simplification du système étudié: le plus simple est la représentation la plus facile sera la spécification du modèle. Ainsi, Jean Bézivin et Olivier Gerbé (Bézivin and Gerbé, 2001) définissent le modèle comme étant « *A simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system* ».

A partir de cet élément central qu'est le modèle, l'IDM définit une première relation reliant le système au modèle: c'est la relation « représentation de » (Atkinson and Kühne 2003; Seidewitz, 2003; Bézivin, 2004).

Pour prendre en considération l'aspect productif du modèle (manipulable sur une machine) Kleppe et al. (2003) proposent la définition suivante: « *Un modèle est une description d'un (ou d'une partie d'un) système écrit dans un langage bien-défini* ».

La notion de langage bien-défini fait appel indirectement au deuxième concept de l'IDM, à savoir le métamodèle.

3.2.2. Le métamodèle et la relation « conforme à »

Tout comme la notion de modèle, le métamodèle est l'objet de discussion au sein de la communauté IDM et plusieurs définitions ont été proposées. Selon (OMG, 2002), « *Un métamodèle est un modèle qui définit un langage pour exprimer un modèle* ». Klepper et al. (2003) donnent une autre définition: « *Un métamodèle est un modèle de spécification pour une classe de systèmes étudiés, où chaque système étudié dans cette classe est lui-même un modèle valide exprimé dans un certain langage de modélisation* ».

Contrairement à une définition courante (Steinberg et al., 2008), le métamodèle n'est pas dans ce contexte un modèle d'un modèle. Cette notion trompeuse du métamodèle (Favre 2004b) est issue des premières heures de la MDA. Le métamodèle peut être plutôt défini comme étant le modèle d'un langage de modélisation. En effet, un métamodèle est un modèle qui définit le langage ou l'ensemble de règles strictes pour exprimer un modèle. Ainsi, tout modèle dans un domaine spécifique est conforme au métamodèle correspondant. Cette définition s'appuie sur la relation suivante entre un modèle et son métamodèle: « un modèle est conforme à un métamodèle ».

La figure 25 (Favre et al., 2006) propose un exemple de carte administrative du territoire français. Cette carte est un modèle prenant en compte la décomposition en régions et départements. Cette décomposition est portée par la légende qui forme le

métamodèle de la carte administrative. Ici la carte est le modèle et la légende est le métamodèle: toute carte administrative est conforme à sa légende.

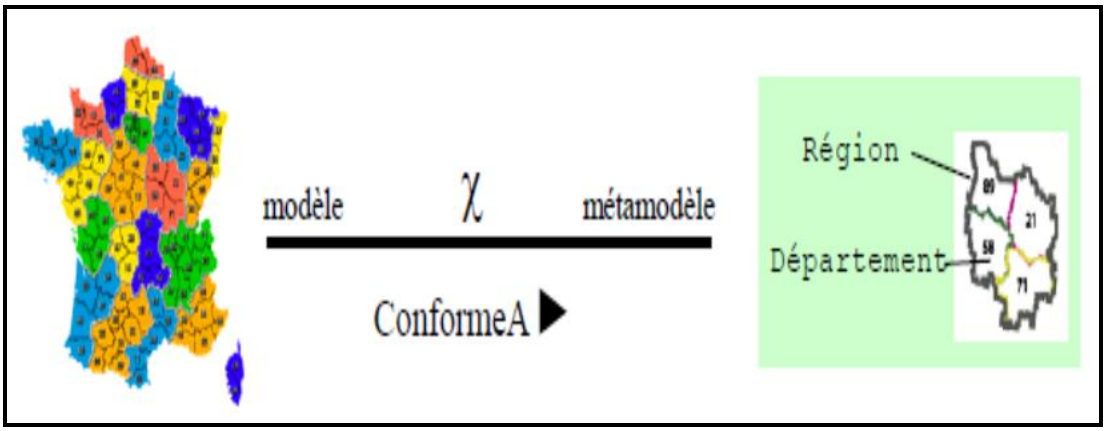

Figure 25. Relation de conformité entre modèle et métamodèle suivant l'exemple des cartes administratives, tiré de (Favre et al., 2006).

Selon les définitions exposées ci-dessus, le métamodèle est un modèle. Il est alors possible de définir le métamodèle auquel il doit se conformer. Il s'agit ici d'un méta-métamodèle. Ce dernier étant un métamodèle, il est également possible de le décrire avec le langage de métamodélisation qu'il définit, il est alors conforme à lui-même.

En se basant sur ces définitions et principes du métamodèle, l'organisation de la modélisation en IDM est représentée par une pyramide de 4 niveaux couramment utilisée par la communauté IDM (figure 26). Le niveau d'abstraction M0 correspond aux systèmes modélisés, M1 représente les modèles représentant ces systèmes, M2 est l'ensemble des métamodèles permettant de définir les modèles de M2 et M3 est le méta-métamodèle.

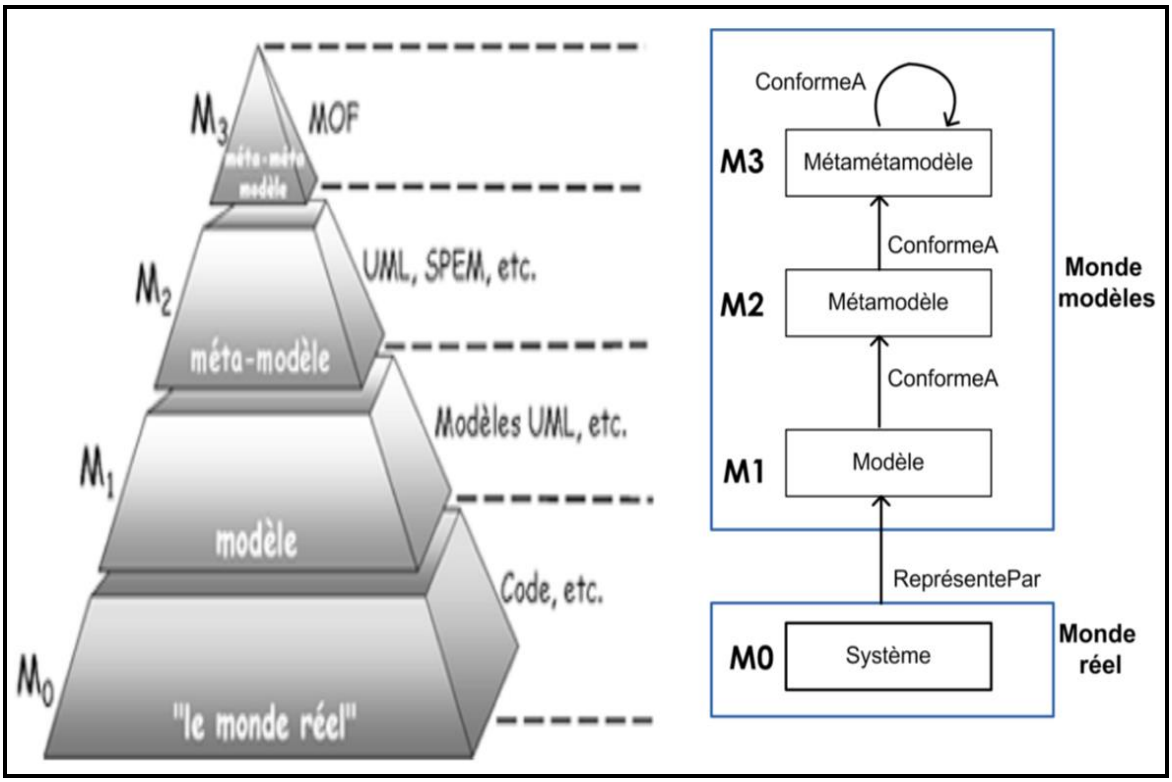

Figure 26. La pyramide d'abstraction de l'IDM à 4 niveaux (source: Hammoudi, 2010).

La relation de conformité a plusieurs formes spécifiques en fonction des espaces techniques étudiés, par exemple:

- ✓ dans l'espace technique des bases de données: on dit qu'une base est conforme au schéma de la base de données;
- ✓ dans l'espace technique XML (« Extensible Markup Language »): un fichier XML est conforme à un schéma ou à une DTD (« Document Type Definition »);
- ✓ dans l'espace technique de la programmation orientée objet: un objet est conforme à sa classe.

La figure 27, issue de (Favre et al., 2006), illustre quatre exemples d'espaces techniques.

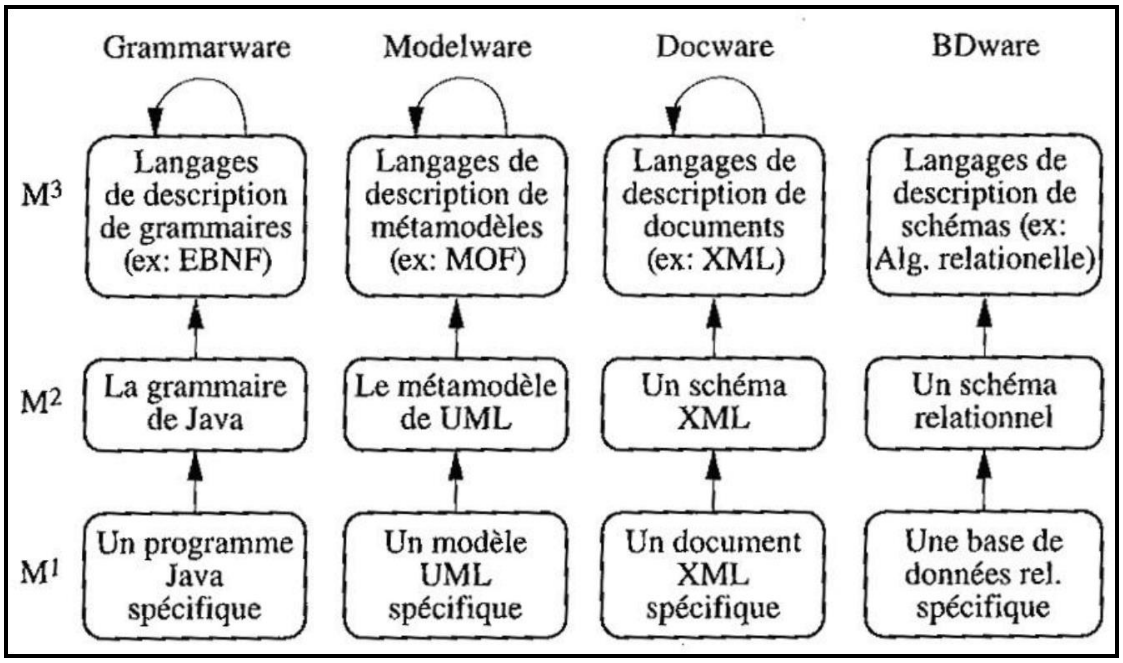

Figure 27. Exemples d'espaces techniques (Favre et al., 2006).

3.2.3. La transformation de modèles et la relation « transformé en »

L'apport clé de l'IDM par rapport aux autres approches de génie logiciel est l'aspect productif des modèles/métamodèles qui ne sont plus « contemplatifs ». Cependant, pour rendre les modèles productifs, il faut pouvoir passer d'un modèle à un autre ou d'un niveau à un autre. Cette notion correspond au troisième principe de l'IDM: « la transformation de modèles » (Selic, 2003). Pour cette notion, on ne trouve pas de définition universelle qui fasse consensus (Rahim and Mansoor, 2008; Lano and Clark, 2008; Iacob et al., 2008). Elle peut être définie, selon Bézivin (2004), comme étant « *la génération d'un ou de plusieurs modèles cibles à partir d'un ou plusieurs modèles sources* ». D'après Favre (2004a), on peut définir la transformation de modèles par la relation « est transformé en». Dans (Diaw et al., 2008), les auteurs résument la transformation de modèles en deux étapes successives: identifier les correspondances de concepts entre modèles cibles et sources au niveau des métamodèles, pour créer des fonctions de transformation qui seront appliquées dans la deuxième étape par un programme spécifique appelé moteur de transformation.

On peut différencier deux types de transformations: endogènes et exogènes (Mens and Van Gorp, 2006). Une transformation est dite endogène lorsque les modèles source et cible sont conformes au même métamodèle. A l'inverse, une transformation est dite exogène lorsque les modèles source et cible sont conformes à deux métamodèles

différents. Pour des systèmes d'information complexes, l'utilisation d'une succession de transformations est recommandée (Hemel et al., 2008).

La transformation doit garantir une interopérabilité (capacité de deux ou plusieurs composants à échanger de l'information et à utiliser l'information échangée) maximale entre différents langages et plates-formes. Elle peut être réalisée pour des métamodèles appartenant à des domaines techniques différents. En plus, une approche de métamodélisation conduit à l'identification d'un domaine spécifique sur lequel le métamodèle va porter d'où l'importance des langages dédiés DSL (« Domain-Specific Language ») et DSML (« Domain-Specific Modeling languages ») qui permettent de fournir aux utilisateurs des concepts propres à leurs métiers et qu'ils maitrisent (Combemale, 2008).

3.3. Notions de langages dans l'IDM

Un DSL (« Domain-Specific Language ») (Van Deursen et al., 2000; Greenfield and Short, 2003; Booch et al., 2004; Mernik et al., 2005) est un langage de haut niveau d'abstraction spécifique à un domaine particulier. Ces langages sont généralement de petite taille et répondent à certaine exigences. En effet, d'après (Sendall and Kosaczynski, 2003), un langage de transformation de modèle doit être efficacement implémenté, expressif et sans ambigüité. Il doit également faciliter la productivité du développeur avec précision, concision et clarté, et fournir les moyens pour combiner des transformations et définir les conditions de leur exécution. Face à la diversité des domaines d'application et la variété des approches de transformation, le nombre d'outils et de DSL ne cesse de croître. Comme exemples, on trouve SmartQVT, ATL (« Atlas Transformation language »), Kermata, XAL (Xion Action Language) (Heitz et al., 2007), VIATRA (Visual Automated Model Transformation), GME, ATOM3, Groove, DOM, etc.

Par rapport à un langage de programmation généraliste, un DSL assure une plus grande concision grâce à sa spécificité. Cela permet d'améliorer la productivité et de réduire les coûts de maintenance (Mernik et al., 2005). De plus, le DSL peut être adapté afin que les experts du domaine puissent l'utiliser sans avoir besoin d'aucune compétence en génie logiciel.

Un langage DSL est décrit par une syntaxe abstraite, une syntaxe concrète et une sémantique (Gray et al., 2007). La syntaxe abstraite exprime de manière structurelle les concepts contenus dans le langage et leurs règles d'association. Cette syntaxe n'est pas

destinée à être manipulée par les utilisateurs finaux dans leur tâche de modélisation. En effet, la conception du modèle est réalisée par la manipulation de la syntaxe concrète qui permet de manipuler les concepts de la syntaxe abstraite. La syntaxe concrète peut être graphique ou textuelle. Dans ce contexte, Gray et al. (2007) font la différence entre les DSLs et les DSMLs (« Domain-Specific Modeling Languages »). Les premiers ont une syntaxe concrète textuelle et les seconds une syntaxe concrète graphique. La sémantique du langage donne le sens des concepts portés par la syntaxe abstraite. Elle définit ses relations et ses contraintes. La sémantique du langage n'est pas systématiquement explicite et sa définition reste une problématique d'actualité (Völter, 2011; Selic, 2012).

4. Conclusion

Dans ce chapitre, nous avons vu deux domaines-clés sur lesquels la thèse s'appuie. Tout d'abord, nous nous sommes intéressés au calcul intensif et à l'utilisation de ses différentes ressources pour la bioinformatique et plus particulièrement pour les applications liées aux biopuces à ADN. En effet, les performances des architectures de calcul à hautes performances sont en évolution permanente et permettent aujourd'hui de résoudre des problèmes biologiques complexes et gourmands en temps de calcul et en espace de stockage. C'est précisément le cas de la conception et l'analyse des biopuces à ADN en écologie microbienne. La diversité des architectures de calcul intensif (SMP, cluster, grille de calcul, etc.) actuellement disponibles et leurs capacités de calcul et de stockage permettent de réduire la complexité et le temps de calcul des algorithmes de sélection de sondes exploratoires. Cette diversité de ressources permet de s'adapter au mieux à la taille et à la complexité de la tâche de conception ou d'analyse des biopuces. Ce point nous ramène au deuxième domaine auquel nous nous intéressons dans ce chapitre: c'est l'Ingénierie Dirigée par les Modèles (IDM). L'IDM est une branche du génie logiciel s'intéressant aux modèles comme outil central dans le processus de développement et de maintenance des systèmes informatiques. Elle s'appuie sur trois concepts fondamentaux: le modèle (« représentation de »), le métamodèle (« conforme à ») et la transformation de modèles (« transformé en »). L'utilisation de l'IDM permet de répondre au besoin de généricité recherché pour nos outils de sélection de sondes exploratoires.

Dans la suite, nous présentons les applications développées durant cette thèse et qui exploitent les technologies de calcul intensif et d'Ingénierie Dirigée par les Modèles. La

première application que nous présentons dans le chapitre suivant, est une nouvelle approche distribuée pour la conception de biopuces phylogénétiques sur grilles de calcul.

Chapitre 4: Sélection de sondes oligonucléotidiques pour biopuces phylogénétiques exploratoires sur grille de calcul

1. Introduction

Comme nous l'avons vu dans les chapitres précédents, les biopuces phylogénétiques oligonucléotidiques (POA) utilisant des sondes exploratoires, constituent l'une des approches les plus simples et efficaces pour la description exhaustive des structures des communautés microbiennes. Cependant, seuls quelques outils de sélection de sondes ont été développés pour assurer la sélection de sondes exploratoires discriminantes. Ici, nous présentons une nouvelle approche distribuée pour la sélection de sondes exploratoires sur grilles de calcul, basée sur la stratégie PhylArray (Militon et al., 2007). Ce logiciel ainsi qu'un autre outil de sélection de sondes exploratoires développé aussi au sein de l'équipe, à savoir KASpOD (Parisot et al., 2012) seront utilisés pour construire une base de données complète de sondes oligonucléotidiques pour l'étude des communautés procaryotiques (Jaziri et al., 2014b). Cette base de données est disponible en accès libre via un site web que nous présentons dans la dernière section de ce chapitre.

2. Algorithme de sélection de sondes oligonucléotidiques pour biopuces phylogénétiques sur grilles de calcul

La plupart des outils existants pour la sélection de sondes adaptées aux biopuces phylogénétiques permettent de sélectionner des sondes ciblant seulement des microorganismes pour lesquels des séquences sont disponibles dans les bases de données internationales. Seuls quelques algorithmes permettent de sélectionner des sondes exploratoires assurant ainsi la mise en évidence de populations microbiennes non encore décrites. PhylArray permet de définir de telles sondes (sensibles et spécifiques) ciblant des groupes de séquences correspondant à un niveau taxonomiques donné (Militon et al., 2007). Cependant, PhylArray peut nécessiter plusieurs jours de calcul pour traiter simplement un groupe contenant un grand nombre de séquences (plusieurs centaines de séquences) (Missaoui, 2009). Pour s'affranchir de cette contrainte, une nouvelle méthode distribuée pour la sélection de sondes oligonucléotidiques régulières et exploratoires à grande échelle sur grilles de calcul est proposée. Cette méthode est la continuité des travaux présentés dans (Missaoui, 2009).

2.1. Algorithme de construction d'une base de données du biomarqueur phylogénétique: gènes exprimant la petite sous unité d'ARN ribosomique

La qualité de cette base de données est essentielle car elle sera le support de la qualité des sondes (exhaustivité des groupes microbiens ciblés et spécificité). Actuellement, les bases de données internationales de séquences nucléiques sont très riches mais présentent des séquences de tailles très variables, de qualités très inégales avec de plus dans certains cas des annotations inexistantes voire erronées. Une utilisation directe de telles bases de données peut conduire à une détermination de sondes complètement inappropriées pour répondre aux questions biologiques posées. Pour ces différentes raisons, nous avons développé un nouvel algorithme qui reconstruit automatiquement une base de données experte. L'algorithme est basé sur plusieurs étapes. Initialement, toutes les séquences des divisions taxonomiques procaryotes (PRO), champignons (FUN), et des échantillons environnementaux (ENV) ont été téléchargées à partir de la base de données de séquences nucléotides de l'EMBL (« European Molecular Biology Laboratory »). Ensuite, nous avons établi une liste de mots clé qui nous a permis d'extraire seulement les séquences des gènes codant la petite sous unité d'ARNr (16S pour les procaryotes et 18S pour les champignons).

Les critères de sélection des séquences ensuite appliqués sont les suivants:

- ✓ La longueur de la séquence est supérieure à 1200 bases.
- ✓ La longueur de la séquence est inférieure à 1600 bases pour les séquences procaryotes et 1800 bases pour les séquences fongiques.
- ✓ Le pourcentage de nucléotides indéterminés (pas {A, C, G, T}) dans la séquence est inférieur à 1%.
- ✓ Le nombre maximal de nucléotides indéterminés consécutifs ne doit pas dépasser 5.

Pour chacune des séquences, l'affiliation est également conservée et extraite à partir du champ descriptif « OC » (« Organism Classification ») des fiches EMBL (figure 28). Ensuite, les séquences sélectionnées (format FASTA) ont été regroupées par genre.

```
ID   AF411064; SV 1; linear; genomic DNA; STD; PRO; 1482 BP.
XX
AC   AF411064;
XX
DT   15-OCT-2001 (Rel. 69, Created)
DT   15-OCT-2001 (Rel. 69, Last updated, Version 1)
XX
DE   Geobacillus anatolicus strain SB 16S ribosomal RNA gene, partial sequence.
XX
KW   .
XX
OS   Geobacillus anatolicus
OC   Bacteria; Firmicutes; Bacilli; Bacillales; Bacillaceae; Geobacillus.
XX
RN   [1]
RP   1-1482
RA   Uysal H., Bakkal S., Erturk D., Bilgin N.;
RT   ;
RL   Submitted (16-AUG-2001) to the INSDC.
RL   Molecular Biology and Genetics, Bogazici University, Bebek, Istanbul 80815,
RL   Turkey
XX
DR   RFAM; RF00177; SSU rRNA bacteria.
DR   RFAM; RF01959; SSU rRNA archaea.
DR   SILVA-SSU; AF411064.
XX
FH   Key             Location/Qualifiers
FH
FT   source          1..1482
FT                   /organism="Geobacillus anatolicus"
FT                   /strain="SB"
FT                   /mol_type="genomic DNA"
FT                   /country="Turkey: Hisaralan, Balikesir"
FT                   /note="type strain; isolated from hydrothermal vent at 98
FT                   degrees Celcius"
FT                   /db_xref="taxon:171243"
FT   rRNA            <1..1482
FT                   /product="16S ribosomal RNA"
XX
SQ   Sequence 1482 BP; 351 A; 371 C; 491 G; 267 T; 2 other;
     gctggcggcg tgcctaatac atgcaagtcg agcggaccga atgagtagct tgctcttgtt        60
     tggtcagcgg cggacgggtg agtaacacgt gggcaacctg cccgcaagac cgggataact       120
     ccgggaaacc ggagctaata ccggataaca ccgaagaccg catggtcttt ggttgaaagg       180
     cggactttgg ctgtcacttg cggatgggcc cgcggcgcat tanctagttg gtgaggtaac       240
     ggttcaccaa ggcaacgatg cgtagccggc ctgagagggt gaccggccac actgggactg       300
     aaacacggcc caaactccta cgggaggcag cagtagggaa tcttccgcaa tggacaaaag       360
     tttgacggag cgacgccgcg tgagcgaaga aggccttcgg gtcgtaaagc tctgttgtga       420
     gggacaaagg agcgccgttt gaacaaggcg gcgcggtgac ggtacctcac gagaaagccc       480
     cggttaacta cgtgccacca gccgcggtaa tacgtagggg gcgagcgttg tccggaatta       540
     ttgggcgtaa agcgcgcgca ggcggttcct taagtctgat gtgaaagccc acggctcaac       600
     cgtggagggt cattggaaac tgggggactt gagtgcagga gaggagagcg gaattccacg       660
     tgtagcggtg aaatgcgtag agatgtggag gaacaccagt ggcgaaggcg gctctctggc       720
     ctgcaactga cgctgaggcg cgaaagcgtg gggagcaaac aggattagat accctggtag       780
     tccacgccgt aaacgatgag tgctaagtgt tagaggggtc acacccttta gtgctgcagc       840
     taacgcgata agcactccgc ctggggagta cggccgcaag gctgaaactc aaaggaattg       900
     acgggggccc gcacaagcgg tggagcatgt ggtttaattc gaagcaacgc gaagaacctt       960
     accaggtctt gacatcccct gacaacccaa gagattgggc gttccccctt cgggggggaca      1020
     gggtgacagg tggtgcatgg ttgtcgtcag ctcgtgtcgt gagatgttgg gttaagtccc      1080
     gcaacgagcg caacccttgc ctctcgttgc cagcattcag ttgggcactc tagagggact      1140
     gccggctaaa agtcggagga aggtggggat gacgtcaaat catcatgccc cttatgacct      1200
     gggctacaca cgtgctacaa tgggcggtac aaagggctgc gaacccgcga gggggagcga      1260
     atcccaaaaa gccgctctca gttcggattg caggctgcaa ctcgcctgca tgaagccgga      1320
     atcgctagta atcgcggatc agcatgccgc ggtgaatacg ttcccgggcc ttgtacacac      1380
     cgcccgtcac accacgagag cttgcaacac ccgaagtcgg tgaggtaacc ctgacgggag      1440
     ccagccgccn aaggtggggc aagtgattgg ggtgaagtcg ta                         1482
//
```

Figure 28. Exemple de fiche EMBL.

L'étape suivante consiste à vérifier que les séquences correspondent bien au brin codant du gène et non à l'autre brin. Pour cela, le programme Blastn (chapitre 2 section 2.2) permet de comparer chaque séquence extraite avec une séquence de référence (la séquence *Bacteroides intestinalis* dont le numéro d'accession[1] est AB437413 pour les procaryotes et la séquence *Blastocladiella emersonii* avec le numéro d'accession EF014366 pour les champignons). Dans le cas où aucune similarité n'est identifiée, la séquence est inversée et complémentée pour correspondre au brin codant du gène.

Par la suite, un regroupement de séquences utilisant le logiciel BlastClust a été effectué sur les séquences de chaque genre pour éliminer les séquences redondantes (conservation d'une seule séquence parmi x séquences présentant 100 % d'identité). Les paramètres utilisés pour cette analyse sont les suivants:

- ✓ -p - F (il s'agit de séquences nucléotidiques);
- ✓ -S 100 (seuil d'identité entre les séquences en pourcentage);
- ✓ -L 1 (longueur de la séquence à couvrir: ici les séquences doivent être 100% similaires sur toute la longueur des séquences);
- ✓ - b F (la couverture telle que spécifiée par -L et - S est requise sur une seule séquence d'une paire).

Enfin, pour chaque groupe, nous avons vérifié l'homogénéité des séquences afin d'éliminer les séquences mal annotées et les séquences chimériques. Pour ce faire, nous avons tout d'abord constitué une base de données de référence contenant une séquence de référence pour chacun des genres. Les séquences de référence ont été déterminées en se basant sur la congruence d'annotation à partir de trois bases de données (NCBI[2], SILVA[3] et RDP[4]). Nous avons par la suite, établi les distances entre toutes les séquences de référence afin de définir pour chaque genre les deux séquences de référence des groupes

[1] Numéro d'accession (en anglais « accession number ») est un identifiant unique attribué à une séquence (protéique ou ADN) lorsqu'elle est déposée dans une banque de données internationale.

[2] www.ncbi.nlm.nih.gov

[3] www.arb-silva.de

[4] rdp.cme.msu.edu

les plus proches. Ainsi, chaque séquence de chaque genre sera testée contre les trois séquences de référence sélectionnées. Si la séquence appartient réellement au genre considéré, elle doit être plus proche de sa propre séquence de référence que des séquences de référence des genres les plus proches. Ici, le score de similarité entre les séquences, reflétant la distance, est calculé à l'aide d'une version modifiée du programme Clustalw (Thompson et al., 1994).

Soit *L* la liste des genres de notre base de données du gène codant la petite sous unité de l'ARNr. Soit *Gi* un genre de *L*, on note par S_{Gi} la séquence de référence de ce genre.

Gj et *Gk* sont les deux genres de *L* les plus proches de *Gi* si et seulement si leurs séquences de référence S_{Gj} et S_{Gk} montrent les distances les plus courtes avec la séquence de référence S_{Gj} du genre testé *Gi*:

$$\forall G \in L \setminus \{Gj, Gk\}:$$
$$d(S_{Gi}, S_{Gj}) \leq d(S_{Gi}, S_G)\ et$$
$$d(S_{Gi}, S_{Gk}) \leq d(S_{Gi}, S_G);$$

Soit *S* une séquence du genre *Gi* de *L*. *S* est retenue dans le groupe homogène de séquences discriminatoires de *Gi* si et seulement si:

$$d(S, S_{Gi}) \leq d(S, S_{Gj});$$
$$d(S, S_{Gi}) \leq d(S, S_{Gk});$$
$$d(S, S_{Gi}) \leq d(S_{Gi}, S_{Gj})\ et$$
$$d(S, S_{Gi}) \leq d(S_{Gi}, S_{Gk}).$$

Nous avons implémenté cet algorithme avec le langage de programmation C++ et nous l'avons utilisé pour construire une base de données des gènes d'ARNr 16S au niveau du genre (la construction d'une base de données fongique des gènes d'ARNr 18S est également en cours). Nous avons ainsi retenu 66 075 séquences procaryotes, dont 1 407 séquences d'archées et 64668 séquences de bactéries, réparties en 2069 genres. Notre algorithme peut facilement être adapté et utilisé pour construire des bases de données de séquences ARNr de haute qualité pour d'autres rangs taxonomiques (famille, ordre, classe, etc.). Il serait ainsi également possible de constituer des bases de données spécifiques d'un environnement donné en considérant l'origine de la séquence (sol, eau douce, système

marins, intestin, etc.). Cependant, l'information sur l'origine des échantillons n'est pas toujours donnée ou clairement indiquée dans les fiches EMBL et nécessite donc une fouille de données minutieuse et complexe.

2.2. Algorithme de sélection de sondes oligonucléotidiques

2.2.1. Implémentation

Nous avons développé et implémenté deux versions de notre algorithme. La première version nommée PhylGrid a été publiée en 2011 (Jaziri et al., 2011) et la deuxième que nous avons appelée PhylGrid2.0 a été publiée en 2014 (Jaziri et al., 2014a). Dans la deuxième version, nous avons optimisé la méthode de parallélisation utilisée ainsi que le script de sélection de sondes, pour plus de performance. Dans ce qui suit, nous présentons seulement la version la plus récente PhylGrid2.0. Nous avons implémenté cette version sous Linux CentOS 5.4 avec les langages de programmation C++ et Perl. Notre programme utilise trois autres programmes: Blast (Altschul et al., 1990), Clustalw-MPI (Li, 2003) et Opal (Wheeler and Kececioglu, 2007). Il utilise quatre principaux paramètres en entrée pour assurer la détermination des sondes:

- ✓ la longueur des sondes,
- ✓ la dégénérescence maximale autorisée d'une sonde consensus,
- ✓ le seuil de spécificité (la valeur minimale de similarité utilisée pour déterminer si la sonde peut s'hybrider avec une séquence non ciblée: la sonde ne doit pas avoir une similarité supérieure ou égale à ce seuil avec une séquence non ciblée),
- ✓ le nombre maximum autorisé d'hybridations croisées.

2.2.2. Stratégie de sélection de sondes

La figure 29 illustre les différentes étapes de l'algorithme liées à la soumission des travaux sur la grille de calcul européenne EGI.

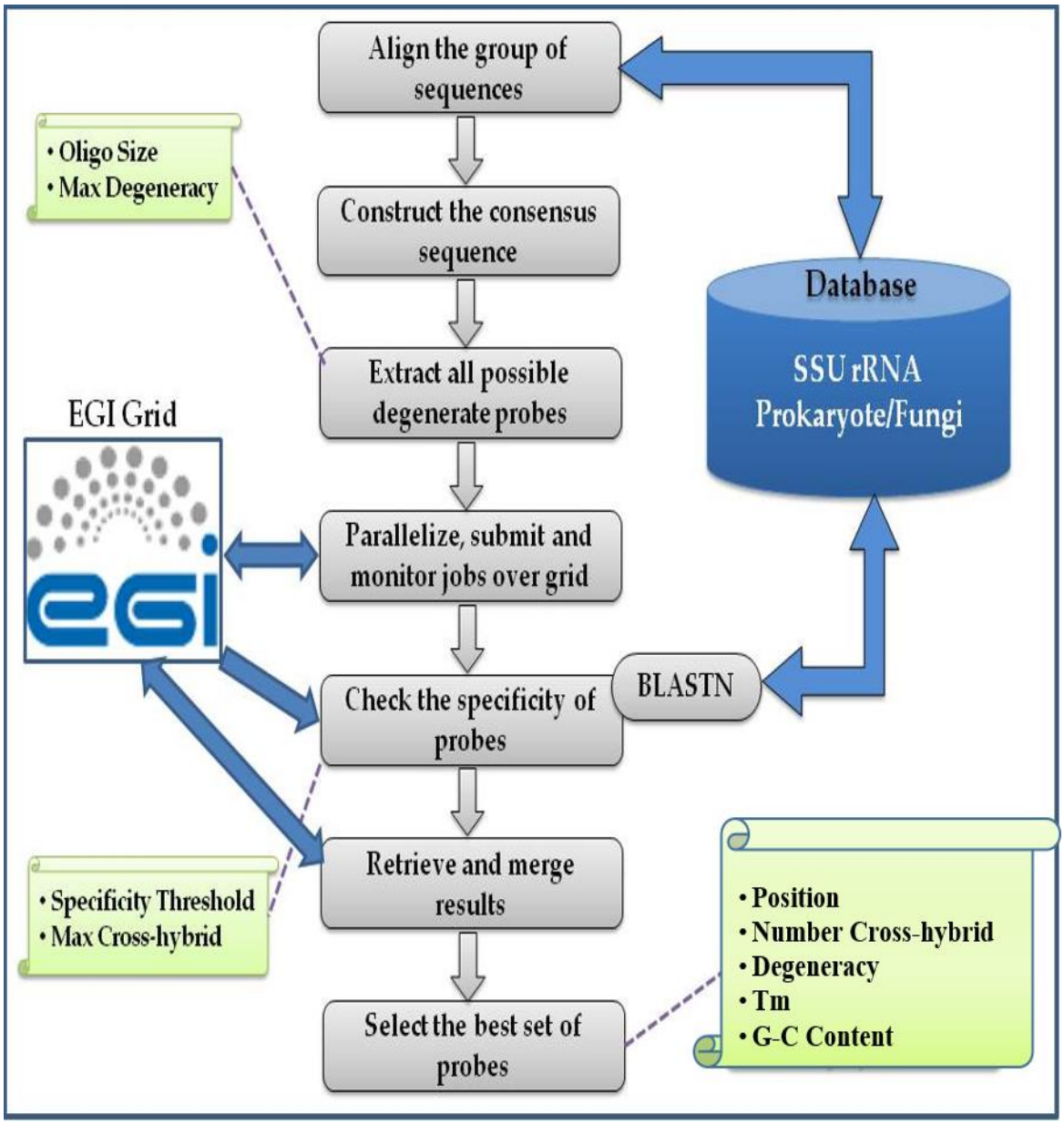

Figure 29. Étapes de l'algorithme de sélection de sondes PhylGrid2.0.

Pour sélectionner des sondes pour un groupe de séquences choisi par l'utilisateur, un alignement multiple des séquences de ce groupe est tout d'abord réalisé. Pour les groupes contenant peu de séquences, Clustalw-MPI (Li, 2003) est utilisé pour réaliser l'alignement. Cependant, pour les groupes contenant un grand nombre de séquences du fait de la complexité de l'alignement et du temps de calcul nécessaire par ClustalW, une stratégie basée sur l'alignement d'alignements a été implémentée. En effet, le temps de calcul nécessaire pour l'alignement multiple d'un grand nombre de séquences avec ClustalW est très important (Missaoui, 2009). Par exemple, environ 8 jours de calcul sont nécessaires pour obtenir l'alignement de 4000 séquences (e.g. le genre *Bacillus* de notre base de

séquences 16S) sur un processeur AMD Opteron de 1.83 Ghz et 128Go de RAM, et d'environ 3h et demi pour effectuer la même tâche avec Clustalw-MPI en utilisant 100 cœurs sur un cluster de 22 nœuds dual-processor Quad Core AMD Opteron à 2,1 GHz.

L'alignement d'alignements de larges groupes de séquences est réalisé en trois étapes. Tout d'abord, nous utilisons BlastClust (avec les paramètres -L .98, S - 98, F -p et -b F) pour obtenir des sous-groupes de séquences homogènes. Les séquences de ces sous-groupes peuvent alors être alignées en utilisant ClustalW-MPI. Le logiciel OPAL (Wheeler and Kececioglu, 2007) réalise ensuite l'alignement d'alignements. Ainsi, les alignements des sous-groupes sont fusionnés pour reconstituer un alignement complet de l'ensemble du groupe. Nous avons testé cette méthode sur différents groupes avec un nombre variable de séquences pour vérifier sa performance en utilisant 100 cœurs de calcul sur le cluster mentionné ci-dessus. Les résultats sont illustrés dans le tableau 4. Par exemple, la performance de cette méthode est 4 fois plus rapide qu'un alignement multiple simple pour le genre bactérien *Bacillus*.

Tableau 4: Comparaison de la performance des méthodes d'alignement de groupes de séquences utilisée dans PhylGrid2.0 et PhylArray (Militon et al., 2007), en utilisant 100 cœurs de calculs sur un cluster de 22 AMD Opterons à 2,1 GHz.

Groupe Aligné	Nombre de séquences	Nombre de sous-groupes	Temps d'alignement (secondes)	
			PhylArray	PhylGrid2.0
Vibrio	1 174	37	2 542	1 247
Bacillus	3 947	58	12 586	3 130

L'alignement créé est ensuite utilisé pour construire une séquence consensus, en se basant sur le code IUPAC des nucléotides dégénérés (Cornish-Bowden, 1985) (tableau 5). L'objectif ici est non seulement d'obtenir une séquence commune qui représente l'ensemble des séquences ciblées du groupe, mais aussi de corriger des erreurs potentielles de séquençage. Par exemple, dans chaque colonne de l'alignement représentant un site moléculaire, si le nombre de nucléotides indéterminés ou absents (« N » ou « - ») est présent dans moins de 50% des séquences de ce groupe, ce sont les nucléotides connus à cette position des autres séquences qui seront retenus. Dans le cas contraire (nombre de

bases indéterminées à un site donné dans plus de 50% des séquences), un gap (« - ») est inséré dans la séquence consensus à cette position.

L'étape suivante de la stratégie de détermination de sondes consiste à extraire toutes les sondes dégénérées possibles de longueur « *l* » (*l* est la longueur de sondes fixée par l'utilisateur) à partir de la séquence consensus du groupe ciblé, en utilisant un pas de 1. Seules les sondes dont la dégénérescence ne dépasse pas la valeur du seuil maximal autorisé de dégénérescence et qui ne contiennent pas de bases indéterminées « N » ou « - », sont extraites.

Tableau 5. Code IUPAC des nucléotides.

IUPAC Code	Définition
A	Adenine
C	Cytosine
G	Guanine
T (ou U)	Thymine (ou Uracil)
R	A ou G
Y	C ou T
S	G ou C
W	A ou T
K	G ou T
M	A ou C
B	C ou G ou T (non A)
D	A ou G ou T (non C)
H	A ou C ou T (non G)
V	A ou C ou G (non T)
N	« any » base
« . » ou « - »	Gap

Par la suite, une parallélisation est réalisée pour distribuer toutes les sondes dégénérées extraites sur « *N* » jobs (*N* est le nombre de jobs défini par l'utilisateur). Pour chaque job,

toutes les sondes possibles, régulières ou exploratoires, sont générées à partir de chaque sonde dégénérée, en utilisant le code IUPAC. La spécificité de ces sondes est ensuite vérifiée contre un fichier en format FASTA contenant seulement les séquences de la base de données de séquences 16S initialement construite qui n'appartiennent pas au groupe ciblé. Pour le test de spécificité, nous utilisons le programme Blastn avec les paramètres suivants:

- ✓ la taille du mot: « *-W 7* »,
- ✓ suppression du filtre des régions de faible complexité: « *–F F* »,
- ✓ l'e-value: « *–e 100* »,
- ✓ « *-S 1* » (indique quel brin nucléotidique à utiliser dans la recherche: brin d'entrée et/ou l'autre brin. Ici 1 pour le brin d'entrée),
- ✓ le nombre total d'alignements (résultats) à afficher: « *-b 20000* ».

Le fichier résultat du programme Blastn est ensuite analysé pour déterminer, pour chaque oligonucléotide, la liste des hybridations croisées potentielles avec un seuil de spécificité supérieur à « *S* » (*S* est le seuil de similarité pour considérer une hybridation croisée, ce seuil est un paramètre saisi par l'utilisateur).

Enfin, toutes les sondes oligonucléotidiques régulières et exploratoires respectant les critères définis par l'utilisateur sont regroupées et sauvegardées dans un fichier résultat. Pour chaque sonde dégénérée ou spécifique, toutes les informations associées sont sauvegardées (e.g. position, dégénérescence, nombre et liste des hybridations croisées, positions des mésappariements, etc.)

2.2.3. Méthode de parallélisation

La sélection des sondes et la vérification de leur spécificité contre une base de données composée de dizaines de milliers de séquences, nécessite un temps de calcul très important. Le logiciel parallèle PhylArray, par exemple, peut nécessiter jusqu'à plusieurs jours de calcul pour sélectionner des sondes pour un seul groupe de séquences (Missaoui, 2009). Afin d'effectuer une sélection plus rapide, notre logiciel s'exécute sur une grille de calcul. L'utilisateur doit d'abord choisir le nombre de jobs à utiliser. La valeur de dégénérescence de chaque sonde respectant les critères de l'utilisateur est ensuite calculée. Une fois cette étape réalisée, toutes les sondes dégénérées extraites sont rassemblées dans un même fichier. Ce fichier doit ensuite être découpé en « *N* » sous-fichiers (*N* est le

nombre de jobs défini par l'utilisateur) en fonction de la valeur de dégénérescence de chaque sonde dégénérée. Tout d'abord, toutes les sondes dégénérées sont classées par ordre décroissant en fonction de leurs valeurs de dégénérescence. La dégénérescence moyenne par sous-fichier est ensuite calculée sur la base de la somme de toutes les valeurs de dégénérescence et du nombre de jobs désiré. Enfin, un algorithme de type « Worst fit » (Johnson, 1974) est utilisé pour que les sous-fichiers créés puissent pratiquement avoir la même valeur de dégénérescence (figure 30). Chaque sous-fichier sera traité par un job soumis sur la grille de calcul européenne EGI.

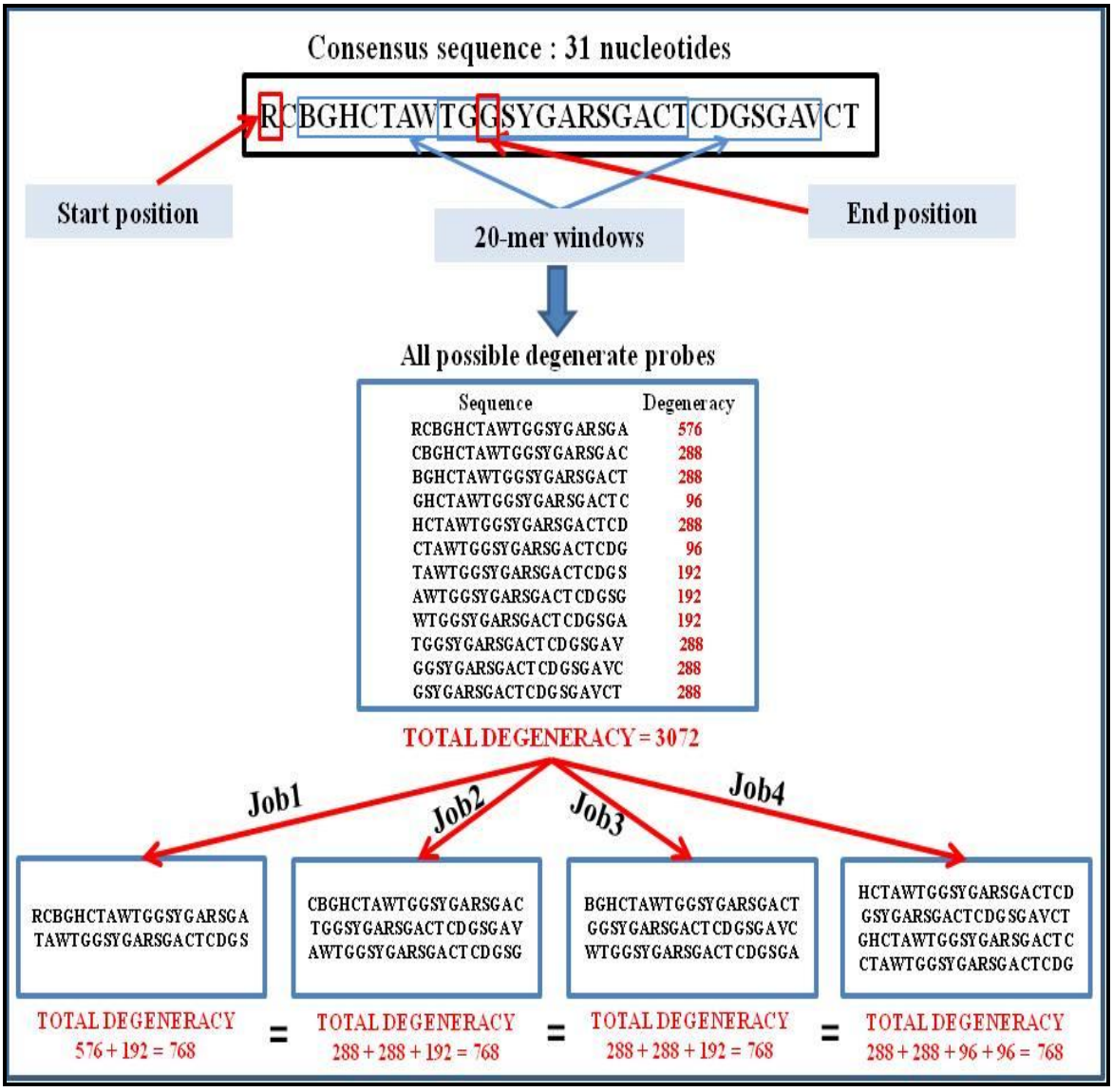

Figure 30. Stratégie de parallélisation pour définir et soumettre les jobs sur grille de calcul.

L'utilisation de la grille de calcul EGI nécessite des autorisations pour accéder ses ressources informatiques. Le CA (« Certificate Authority »[5]) (e.g. l'autorité de certification GRID2-FR du CNRS en France) fournit, sur demande et après vérification, à chaque utilisateur, un certificat à durée de validité limitée composé d'un couple de clés: une publique et une privée. Les deux clés sont générées à partir d'un seul fichier au format PKCS12 (« Public Key Cryptographic Standards ») avec l'extension « .p12 ». Ce format est utilisé pour stocker la clé privée et le certificat de clé publique correspondant en les protégeant par un mot de passe.

Les commandes suivantes permettent de générer les deux clés publique et privée pour une utilisation de la grille:

- ✓ Pour obtenir la clé publique:

```
openssl pkcs12 -in mycert.p12 -clcerts -nokeys –out ~/.globus /usercert.pem
```

- ✓ Pour obtenir la clé privée:

```
openssl pkcs12 -in mycert.p12 -nocerts -out ~/.globus/userkey.pem
```

Les deux fichiers d'extension « .pem » ainsi générés seront ensuite nécessaires pour créer un proxy permettant l'authentification de l'utilisateur et l'autoriser à utiliser toutes les machines de la grille durant une certaine période. La création du proxy se fait à l'aide de la commande suivante:

```
voms-proxy-init --voms <VOName>
```

[5] L'autorité de certification (en anglais « Certificate Authority ») est l'infrastructure qui assure la délivrance des certificats électroniques utilisés comme mode de d'authentification pour accéder à l'ensemble des services de la grille.

Nous avons utilisé pour les développements réalisés les ressources de l'organisation virtuelle (VO) biomed[6]. C'est une VO à grande échelle, internationale et multi disciplinaire. Biomed est opérationnelle sur l'infrastructure de la grille EGI. Elle regroupe les utilisateurs de la grille EGI dans le secteur des sciences de la vie.

Pour les conditions d'exécution, la base de données est copiée sur des serveurs de stockage SEs (« Storage Element ») de la grille accessibles par tous les jobs relatifs à une sélection de sondes (voir chapitre 3 section 2.6 pour plus de détail sur les SEs et les autres éléments de la grille utilisés dans ce paragraphe). Le programme Blastn nécessaire à la recherche de spécificité est également copié sur le « Worker Node » à partir de son LFN (« Logical File Name ») sur la grille. En ce qui concerne la soumission, il est important de ne pas surcharger le système de gestion des jobs (WMS). Pour cela, notre programme attend jusqu'à ce que chaque job soit entièrement associé à un élément de calcul (CE) de la grille EGI avant de soumettre le job suivant. Pour chaque job, les fichiers de configuration suivants sont nécessaires pour une soumission sur la grille EGI:

- ✓ **Fichier JDL:** chaque job a besoin d'un fichier JDL (« Job Description Language ») qui sera soumis à la grille et dans lequel le job est décrit: l'exécutable et les paramètres utilisés sont spécifiés. La figure 31 montre un exemple de fichier JDL.

```
PerusalFileEnable = true;
PerusalTimeInterval = 120;
Executable="/bin/bash";
RetryCount = 3;
Arguments = "PhylGrid_2012.1210.0530.00_1.sh";
StdOutput = "PhylGrid_2012.1210.0530.00_1.out";
StdError = "PhylGrid_2012.1210.0530.00_1.err";
InputSandBox = {"CheckProbes.pl","PhylGrid_2012.1210.0530.00_1.sh"};
OutputSandbox = {"PhylGrid_2012.1210.0530.00_1.out",
                 "PhylGrid_2012.1210.0530.00_1.err",
                 "result_Propionibacterium1.txt"};
```

Figure 31. Exemple de fichier JDL pour un job de détermination de sondes pour le genre *Propionibacterium*.

[6] http://lsgc.org/en/Biomed:home

- ✓ **Fichiers de script:** ces fichiers décrivent les traitements élémentaires qui seront exécutées sur la grille. Ils contiennent donc les commandes du système d'exploitation et les scripts nécessaires pour effectuer l'étape de sélection de sondes correspondante à la vérification de la spécificité, à partir d'un fichier contenant les sondes dégénérées. Pendant l'exécution, deux fichiers sont copiés sur le CE sur lequel le job est exécuté: le premier contient la base de données de séquences contenant les séquences pour évaluer les hybridations croisées potentielles et le second contient les sondes dégénérées à traiter par ce job. La figure 32 contient un exemple de script shell SH[7] pour un job créé et soumis à la grille par notre programme pour sélectionner des sondes ciblant le genre *Propionibacterium*.

```
rm -rf blast/
mkdir blast
cd blast/
lcg-cp lfn:/grid/biomed/phylgrid/tmp/blast.tgz file:./blast.tgz
tar xvfz blast.tgz
cd ..
lcg-cp lfn:/grid/biomed/phylgrid/tmp/2012.1210.0530.00_arn.tar
file:./2012.1210.0530.00_arn.tar
rm -f arn*
tar xvf 2012.1210.0530.00_arn.tar
rm -f Propionibacterium_Oligos1
lcg-cp lfn:/grid/biomed/phylgrid/tmp/Propionibacterium_Oligos1
file:./Propionibacterium_O    ligos1
perl CheckProbes.pl Propionibacterium 1 25 0.92 100
rm -f 2012.1210.0530.00_arn.tar
rm -f Propionibacterium_Oligos1
rm -f arn.txt*
rm -rf blast\
rm -rf CheckProbes.pl
rm -f Propionibacterium_Oligos1
```

Figure 32. Exemple de script SH pour un job de détermination de sondes pour le genre *Propionibacterium*.

[7] Un script shell se présente sous la forme d'un fichier contenant une ou plusieurs commandes qui seront exécutées de manière séquentielle.

En outre, nous avons développé des scripts assurant le suivi des différents jobs, pour lancer une nouvelle soumission en cas de défaillance et ainsi d'améliorer la fiabilité de notre logiciel tout en optimisant notre utilisation de la grille de calcul. Trois cas peuvent donc être distingués:

- ✓ La soumission du job a échoué: ce job est alors resoumis à la grille EGI. Cette action est répétée tant que la soumission n'est pas encore réussie.

- ✓ Le job est soumis à la grille avec succès mais a échoué lorsqu'il est exécuté: un nouveau job est créé avec les mêmes paramètres et est ensuite soumis à la grille.

- ✓ Le job est soumis avec succès et est exécuté avec succès, mais les autres jobs ne sont pas encore terminés: le programme attend la fin de l'exécution avec succès de tous les jobs, puis récupère et fusionne tous les résultats des jobs dans un seul fichier résultat.

Enfin, notre programme a été conçu pour être extensible en séparant les jobs des différentes étapes de sélection de sondes. En effet, il crée un identifiant unique pour chaque design de sondes ciblant un groupe donné exécuté à un moment précis, ce qui permet de séparer et distinguer les différents jobs soumis à la grille même si plusieurs sélections de sondes sont lancées simultanément.

2.3. Résultats de la parallélisation et de la soumission de jobs sur la grille de calcul européenne

Dans cette section, nous présentons les résultats obtenus par notre logiciel sur des données réelles. Nous montrons aussi la performance de notre méthode de parallélisation par rapport au programme initial PhylArray (Militon et al., 2007).

2.3.1. Résultats de la méthode d'équilibrage de charge

Pour tester l'efficacité de notre méthode d'équilibrage de charge décrite précédemment, nous l'avons comparé à la méthode initiale utilisée dans le programme original PhylArray fonctionnant sur un cluster de calcul. Ainsi, pour distribuer le calcul sur *N* jobs, PhylArray divise le nombre de sondes (répondant aux critères de l'utilisateur) provenant d'une séquence consensus en *N* sous-groupes. Chaque sous-groupe est ensuite traité sur un cœur de calcul. La dégénérescence de chaque sonde n'est donc pas considérée.

Les tests de comparaison ont été effectués sur des ensembles de données réelles, en utilisant respectivement 16 jobs pour sélectionner des sondes ciblant le groupe de séquences du genre bactérien *Citrobacter* (figure 33), 8 pour sélectionner des sondes pour le genre bactérien *Haemophilus* (figure 34), et enfin 4 jobs pour sélectionner des sondes pour les genres bactériens: *Citrobacter, Eubacterium* et *Haemophilus* (tableau 6). Le fait de considérer la dégénérescence des sondes pour réaliser l'équilibrage des calculs sur les différents processeurs assure une meilleure efficacité du traitement. Par exemple, comme le montre le tableau 6, l'écart type entre les charges des différents jobs créés avec notre logiciel est très faible (0,5 sonde) par rapport à l'écart type très élevé obtenu lors de l'utilisation de PhylArray (18 647,85 sondes).

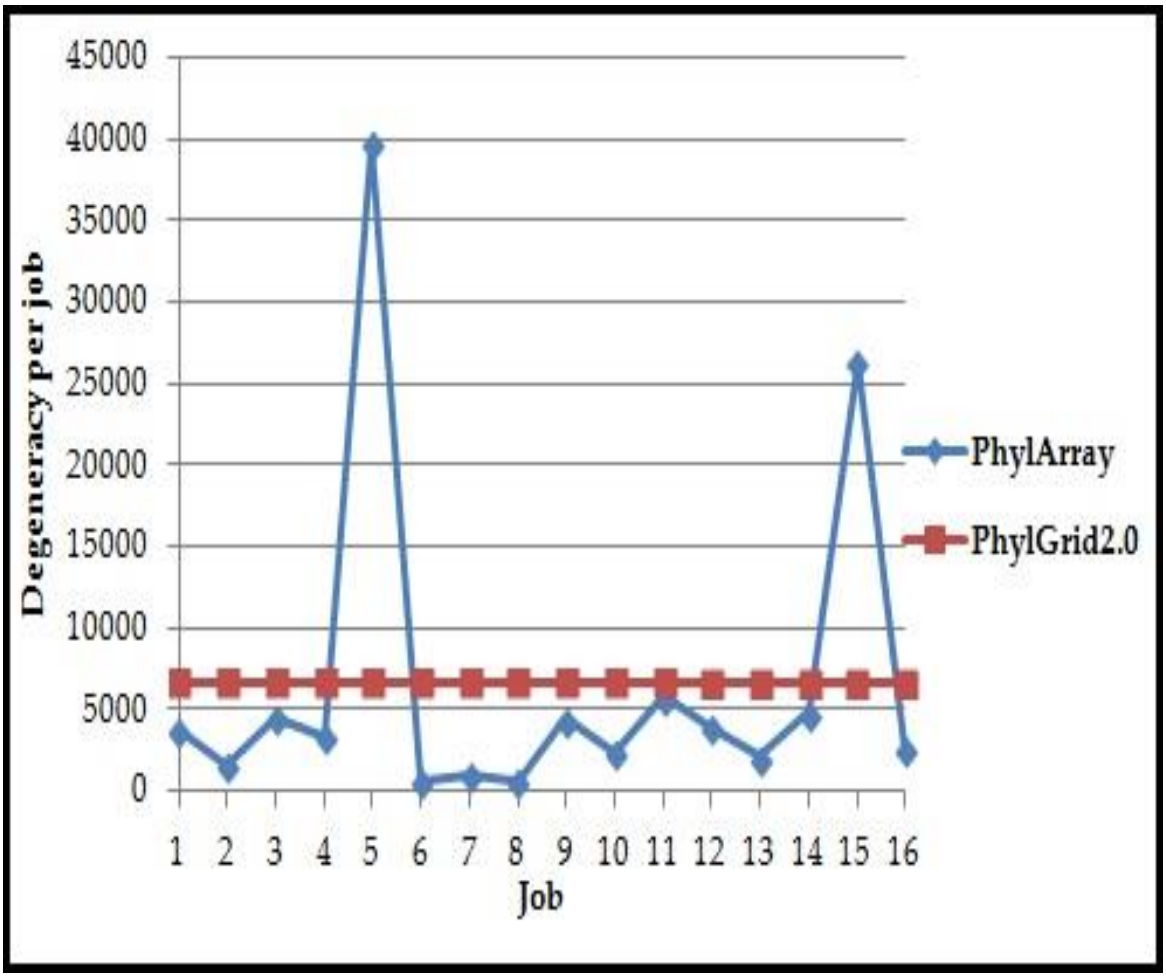

Figure 33. Comparaison de notre nouvelle méthode d'équilibrage de charge avec celle utilisée initialement dans PhylArray (Militon et al., 2007), en utilisant 16 cœurs de calcul pour sélectionner des sondes ciblant le genre *Citrobacter*.

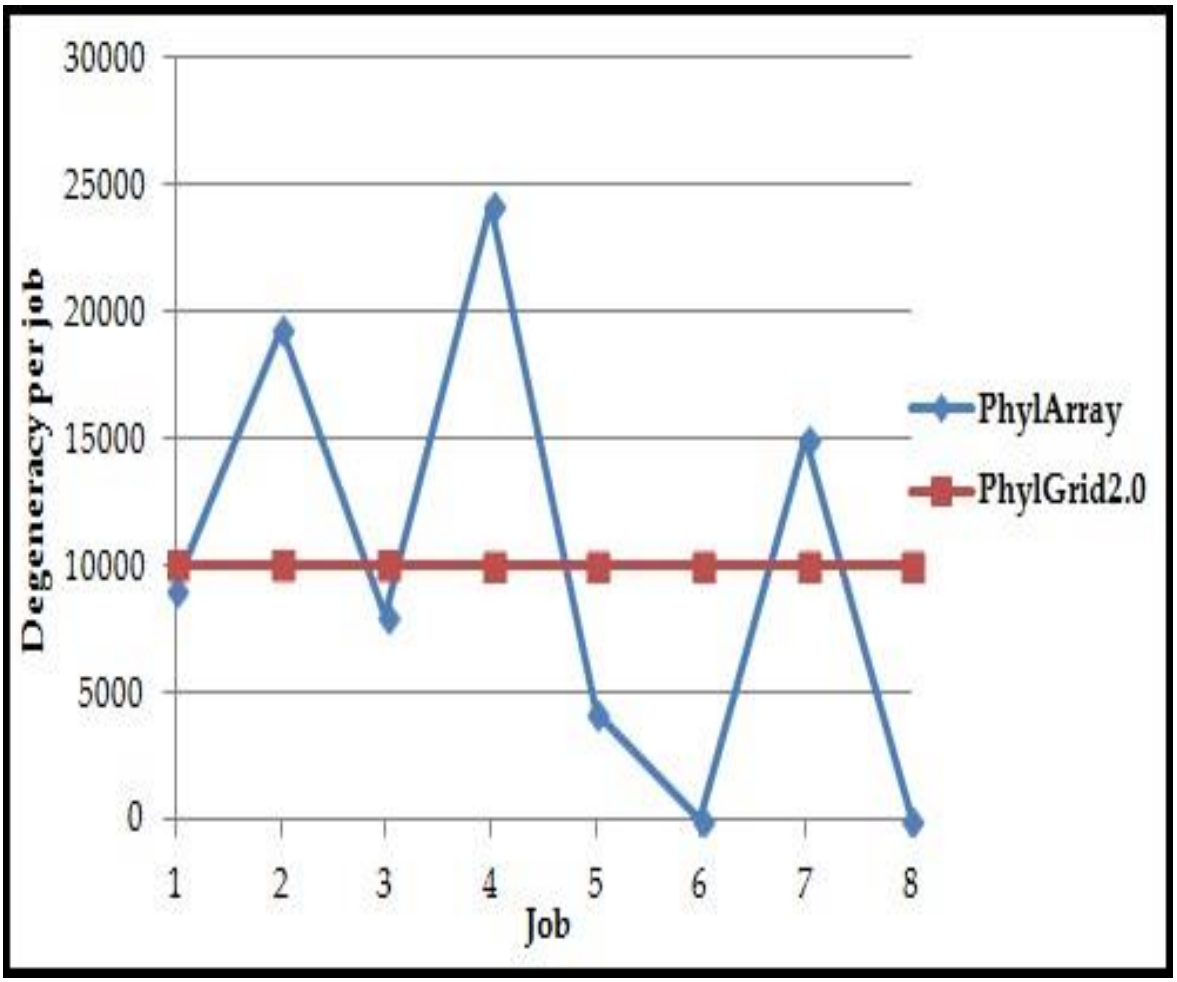

Figure 34. Comparaison de notre nouvelle méthode d'équilibrage de charge avec celle initialement utilisée dans PhylArray (Militon et al., 2007), en utilisant 8 cœurs de calcul pour sélectionner des sondes ciblant le genre *Haemophilus*.

Tableau 6: Comparaison de notre nouvelle méthode d'équilibrage de charge avec celle initialement développée dans PhylArray (Militon et al., 2007), en utilisant 4 cœurs de calcul pour sélectionner des sondes ciblant 3 genres procaryotes: *Citrobacter*, *Eubacterium* et *Haemophilus*.

Genre	***Citrobacter***		***Eubacterium***		***Haemophilus***	
Logiciel	**PhylArray**	**PhylGrid2.0**	**PhylArray**	**PhylGrid2.0**	**PhylArray**	**PhylGrid2.0**
Dégénérescence moyenne	26 722,75	26 722,75	37 132,25	37 132,25	20 100,75	20 100,75
Dégénérescence coeur1	13 068	26 723	41 435	37 133	28 600	20 101
Dégénérescence coeur2	41 782	26 723	43 466	37 132	32 335	20 101
Dégénérescence coeur3	16 381	26 723	10 273	37 132	4 314	20 101
Dégénérescence coeur4	35 660	26 722	53 355	37 132	15 154	20 100
Ecart type	14 142,836	**0,5**	18 647,85	**0,5**	12 853,09	**0,5**

2.3.2. Résultats du déploiement de PhylGrid2.0 sur la grille de calcul européenne

Notre logiciel permet aux utilisateurs de soumettre des jobs parallèles à la grille de calcul EGI dans le but de sélectionner des sondes oligonucléotidiques pour biopuces phylogénétiques. Pour tester les performances de notre approche, nous avons réalisé une détermination de sondes pour 10 groupes de séquences simultanément. Les groupes de séquences ciblés sont les genres procaryotes suivants: *Eubacterium, Citrobacter, Propionibacterium, Arcanobacterium, Campylobacter, Bacteriovorax, Kaistobacter, Neisseria, Haemophilus* et *Riemerella*. Nous avons utilisé les paramètres suivants:

- ✓ Longueur de la sonde = 25-mers;
- ✓ Spécificité = seuil de 0,88 (la sonde ne doit pas avoir une similarité supérieure ou égale à 88%, avec une séquence non ciblée);
- ✓ Nombre maximum d'hybridations croisées = 100;
- ✓ Dégénérescence maximale = 2000.

La réalisation de ces calculs sur un processeur avec un seul cœur de calcul nécessite plus de 8 mois. Nous avons soumis ces calculs à la grille EGI en utilisant un total de 600 jobs. Ce test a été réalisé 5 fois et le résultat médian en termes de temps de calcul a été considéré. Nous avons obtenu tous les résultats après moins de 55 heures incluant l'attente liée à la soumission et à l'exécution des jobs. Les résultats sont illustrés dans la figure 35.

La performance obtenue avec 600 jobs soumis est donc 106 fois plus rapide en temps de calcul malgré les temps d'attente des soumissions sur la grille. En effet, les jobs soumis à une grille de calcul peuvent passer des heures dans les files d'attente. L'indisponibilité de certaines ressources de la grille telles qu'un élément de calcul ou un élément de stockage peut également causer la perte ou le blocage de certains jobs. Cette étape peut bien sûr augmenter le temps de calcul global de notre logiciel qui resoumet toutefois les jobs perdus ou ayant échoué. Par exemple, dans la fenêtre de temps entre 20 et 30 heures de la figure 35, on peut voir une petite diminution de la vitesse de retour des résultats. Cela est dû à l'importance du nombre de resoumissions des jobs ayant échoués dans cette période. Ces problèmes de soumission et d'exécution des jobs sur la grille peuvent expliquer les différences observés pour les 5 réplicas (tableau 7).

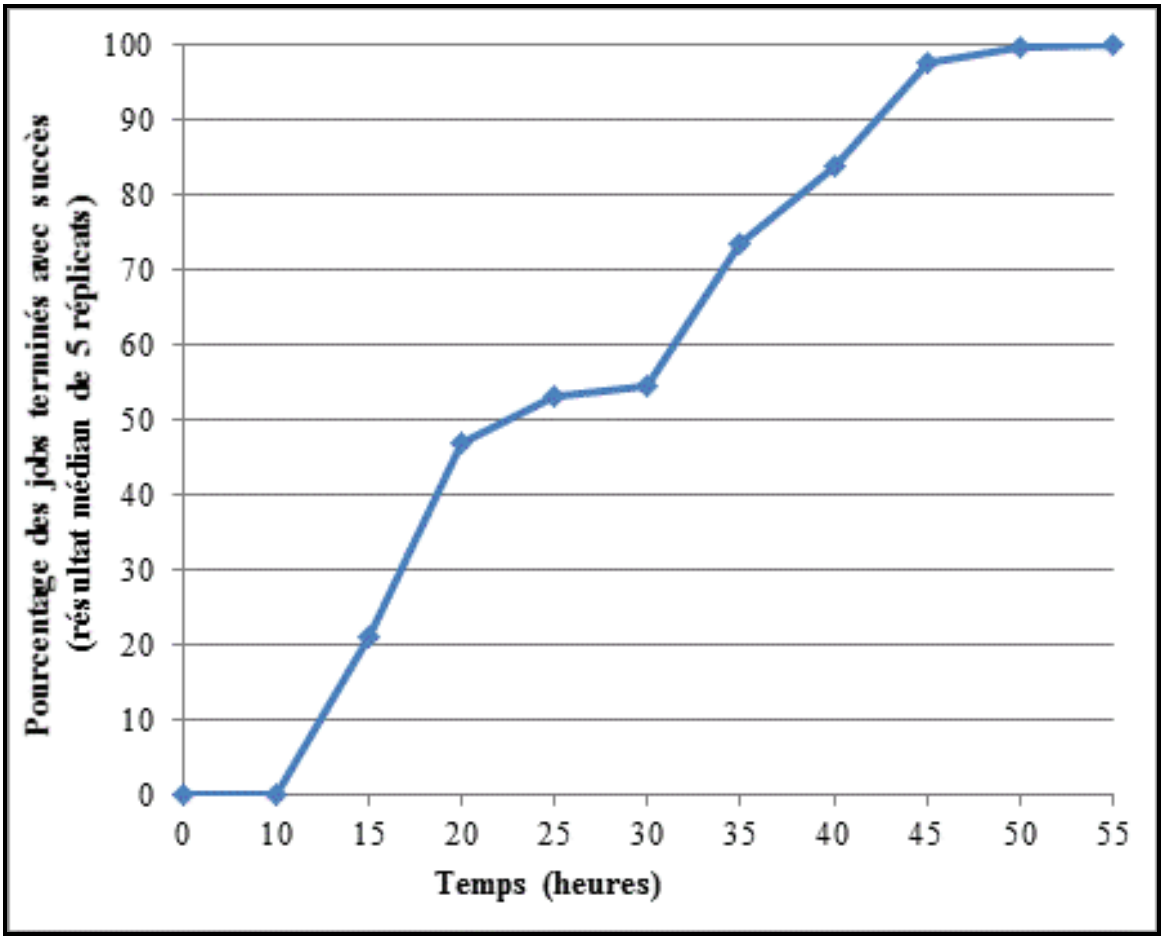

Figure 35. Temps d'exécution médian de sélection de sondes pour 10 genres procaryotes en utilisant 600 jobs soumis à la grille de calcul EGI (résultat médian des 5 réplicats soumis à la grille).

Tableau 7: Temps d'exécution de sélection de sondes pour 10 genres procaryotes en utilisant 600 jobs soumis à la grille EGI (résultat pour les 5 réplicats soumis à la grille).

Temps (heures) (avec l'attente liée à la soumission des calculs)				
Médiane	Moyenne	Min	Max	Ecart type
55	64	50	90	16,35

2.4. Discussion

Dans ce travail, nous avons montré qu'il est possible de sélectionner des sondes à grande échelle sur une infrastructure de grille de calcul avec des gains de performance significatifs. Notre logiciel permet de sélectionner des milliers de sondes en se basant sur la stratégie PhylArray qui assure une bonne sensibilité et spécificité des sondes (Militon et al., 2007). La sélection de sondes régulières et exploratoires à grande échelle, est une tâche complexe et très coûteuse en temps de calcul (jusqu'à plusieurs jours de calcul pour un groupe contenant plusieurs centaines de séquences (Missaoui, 2009)). Pour cela, nous avons développé une méthode de parallélisation efficace qui permet de répartir équitablement le traitement d'une tâche de sélection de sondes sur plusieurs jobs. Les résultats expérimentaux obtenus ont montré que le déploiement de notre logiciel sur la grille de calcul européenne EGI a augmenté de manière significative ses performances pouvant atteindre un speedup de 106 avec 600 jobs de calcul soumis à la grille EGI et ce malgré le temps de latence de la grille.

De plus, nous avons développé des scripts de suivi des soumissions pour améliorer la fiabilité et l'efficacité de notre logiciel sur la grille. Néanmoins, la performance de PhylGrid2.0 dépend de la disponibilité des ressources sur la grille, mais aussi du nombre et de la taille des groupes de séquences ciblés. En effet, pour les petits groupes de séquences (quelques dizaines de séquences maximum), le temps d'attente et de transfert des données sur la grille peut largement dépasser le temps de calcul et la grille de calcul n'est pas adaptée à ces calculs qui reste intensifs, mais ne sont pas à l'échelle de la grille. Pour ce type de calculs, l'utilisation d'un multiprocesseur ou d'un cluster de calcul est plus adéquate que l'utilisation d'une grille de calcul. D'autre part, la stratégie de sélection de sondes exploratoires peut également être améliorée pour assurer une meilleure qualité de sondes.

Une base de données de qualité étant la condition sine qua non pour la détermination de sondes spécifiques, nous avons aussi développé un nouvel algorithme pour la construction des bases de données de séquences des gènes exprimant la petite sous unité de l'ARN ribosomique. Nous avons testé cet algorithme pour créer une base de données de séquences de l'ARNr 16S au niveau du genre. Cette base contient plus de 66 000 séquences représentant 2 069 genres procaryotes.

En plus des sondes régulières ciblant des séquences existantes dans les banques de données génomiques, notre programme permet la sélection de sondes exploratoires ciblant des séquences inconnues appartenant potentiellement à des nouvelles espèces non encore découvertes. Cette fonctionnalité permet d'explorer la grande majorité des microorganismes qui reste jusqu'à lors inconnue. Cependant, certaines sondes exploratoires sélectionnées peuvent, malgré leur importance, causer des problèmes d'hybridation croisées. Cette information de croisement doit être considérée lors des interprétations. Finalement, la qualité des sondes sélectionnées peut encore être améliorée par l'introduction d'autres critères de sélection tels que la présence d'homopolymères ou la possible formation de structures secondaires qui peuvent empêcher l'hybridation sonde/cible.

3. PhylOPDb: une base de données de sondes ciblant le gène de l'ARN ribosomique 16S pour l'identification des procaryotes

Bien que de nombreuses sondes oligonucléotidiques aient été rapportées, les collections disponibles de sondes oligonucléotidiques ciblant l'ARNr sont rares. Par exemple, la base de données de sondes oligonucléotides OPD (« Oligonucleotide Probe Database ») (Alm et al., 1996) a été proposée en 1996 pour collecter des sondes oligonucléotidiques phylogénétiques testées biologiquement. Le dernier ensemble de données d'OPD liste 96 amorces et sondes ciblant les gènes de la petite et la grande sous-unité de l'ARNr. Cependant, OPD n'a pas été actualisée depuis 1997 et n'est plus disponible en ligne. Plus récemment, « probeBase » (Loy et al., 2007), une ressource en ligne pour les sondes oligonucléotidiques publiées ciblant l'ARNr, et les informations associées à ces sondes, a été créée en 2002. Elle comprend actuellement 2 788 sondes (état avril 2014).

Ici, nous présentons PhylOPDb (« Phylogenetic Oligonucleotide Probe Database ») une ressource web pour une base de données exhaustive de sondes oligonucléotidiques phylogénétiques ciblant les séquences du gène ARNr 16S au niveau du genre. Les sondes de PhylOPDb peuvent être utilisées pour différentes techniques de biologie moléculaire: biopuces à ADN, PCR (« Polymerase Chain Reaction »), technique FISH (« Fluorescent In Situ Hybridation »), capture ciblée de gènes ou l'affiliation *in silico* afin d'identifier les populations procaryotiques au sein d'environnements complexes.

3.1. Création de PhylOPDb

Pour développer PhylOPDb (Jaziri et al., 2014b), nous avons utilisé deux logiciels de sélection de sondes à grande échelle développés dans l'équipe, à savoir PhylGrid2.0 (Jaziri et al., 2014a) et KASpOD (Parisot et al., 2012). Ces deux logiciels ont la particularité de pouvoir définir des sondes exploratoires.

Avec PhylGrid2.0, nous avons utilisé notre base de données experte décrite précédemment (section 2.1) afin de déterminer un total de 3 553 975 sondes de 25-mers. Cette longueur assure une spécificité très élevée tout en conservant une bonne sensibilité (Lipshutz et al., 1999; Militon et al., 2007). Des tests de couverture (pourcentage de séquences cibles reconnues par une sonde) et de spécificité ont été ensuite effectués contre la base de données d'entrée avec un seuil de spécificité de 92%, considérant ainsi qu'avec moins de 3 mésappariements une séquence non ciblée peut potentiellement entraîner une hybridation croisées. Le nombre de mésappariements autorisé a été choisi en tenant compte de l'effet de la déstabilisation sur le couple sonde-cible et de la perte de signal (Loy et al., 2002). Tout en essayant d'assurer la meilleure spécificité et une dégénérescence minimale, seulement cinq sondes dégénérées non chevauchantes ont été sélectionnées pour chaque genre, au cours de cette étape. De ce fait, même si certaines sondes présentent des hybridations croisées, l'analyse simultanée des cinq sondes dégénérées évitera des interprétations erronées des données d'hybridation. Finalement, un ensemble de 19 874 sondes 25-mers correspondant à 10 320 sondes dégénérées ciblant 2 096 genres procaryotes ont été obtenues.

En ce qui concerne les sondes sélectionnées à l'aide de KASpOD, la sélection était basée sur la base de données Greengenes (McDonald et al., 2012). La version de mai 2011 contenant 406 997 séquences procaryotes de l'ARNr 16S a été téléchargée. Puis, seules les séquences assignées à un genre ont été retenues, ce qui correspond à 310 575 séquences. Ces séquences ont été ensuite triées par genre et stockées dans des fichiers FASTA. Pour chaque genre, une étape de clustering a été effectuée à un seuil de 100% d'identité en utilisant l'outil CD-HIT (Li and Godzik, 2006) afin d'éliminer les séquences redondantes. Puis, seules les séquences de haute qualité ont été retenues sur la base des critères suivants:

- ✓ La longueur des séquences doit être supérieure à 1200 nucléotides.
- ✓ Les séquences doivent contenir moins de 1% de nucléotides indéterminés.

Un clustering de l'ensemble des séquences retenues à des seuils d'identité élevés (99%, 98% et 97%), couplé à une vérification manuelle, a permis ensuite la suppression des séquences potentiellement mal affectées à un genre donné. De plus, certains genres ne pouvant être discriminés sur la base de l'analyse du gène exprimant l'ARNr 16S, ont été regroupés. 252 183 séquences d'ARNr 16S ont été retenues. Ces séquences ont été utilisées par KASpOD pour effectuer la sélection de sondes (voir chapitre 2 section 2.2.2 pour plus de détail sur la sélection de sondes avec KASpOD). Un total de 3 242 105 sondes 25-mers ont été alors sélectionnées pour 1 295 genres procaryotes. Concernant la spécificité, toute séquence présentant au maximum 2 mésappariements avec une sonde sera considérée comme pouvant provoquer une hybridation croisée. L'ensemble minimal de sondes non chevauchantes caractéristique de chaque genre, assurant la meilleure couverture et la meilleure spécificité possible, a été ensuite déterminé: i) les sondes non chevauchantes ont tout d'abord été sélectionnées parmi les sondes n'entraînant aucune hybridation croisée potentielle avec d'autres genres non ciblés. ii) Dans les cas où certains groupes ne sont pas ciblés par au moins trois sondes, KASpOD sélectionne des sondes supplémentaires pour ces groupes en autorisant des d'hybridations croisées potentielles.

Finalement, et après la suppression des sondes redondantes avec celles sélectionnées avec PhylGrid2.0, nous avons retenu 54 129 sondes oligonucléotidiques qui ont été ajoutées à PhylOPDb. Au total, PhylOPDb est actuellement composé de 74 003 sondes 25-mers ciblant un total de 2 178 genres procaryotes (bactéries et archées).

3.2. Développement d'une interface web pour PhylOPDb

Afin de rendre facilement accessibles et téléchargeable toutes les sondes oligonucléotidiques phylogénétiques contenues dans PhylOPDb, nous avons développé une interface web. Le modèle entité-association représentant la base de données de notre application web est représenté dans la figure 36.

Notre site web est accessible librement et sans inscription préalable sur http://g2im.u-clermont1.fr/PhylOPDb/. Il a été développé en utilisant PHP, HTML5, CSS3, JavaScript, jQuery et MySQL. Il fournit une interface web conviviale et facile à utiliser (figure 37) pour parcourir notre ensemble de sondes régulières et exploratoires ciblant l'ARNr 16S. PhylOPDb sera mise à jour chaque année pour l'ajout de nouvelles sondes, la suppression des sondes obsolètes et la mise à jours de la couverture et de la spécificité des sondes existantes.

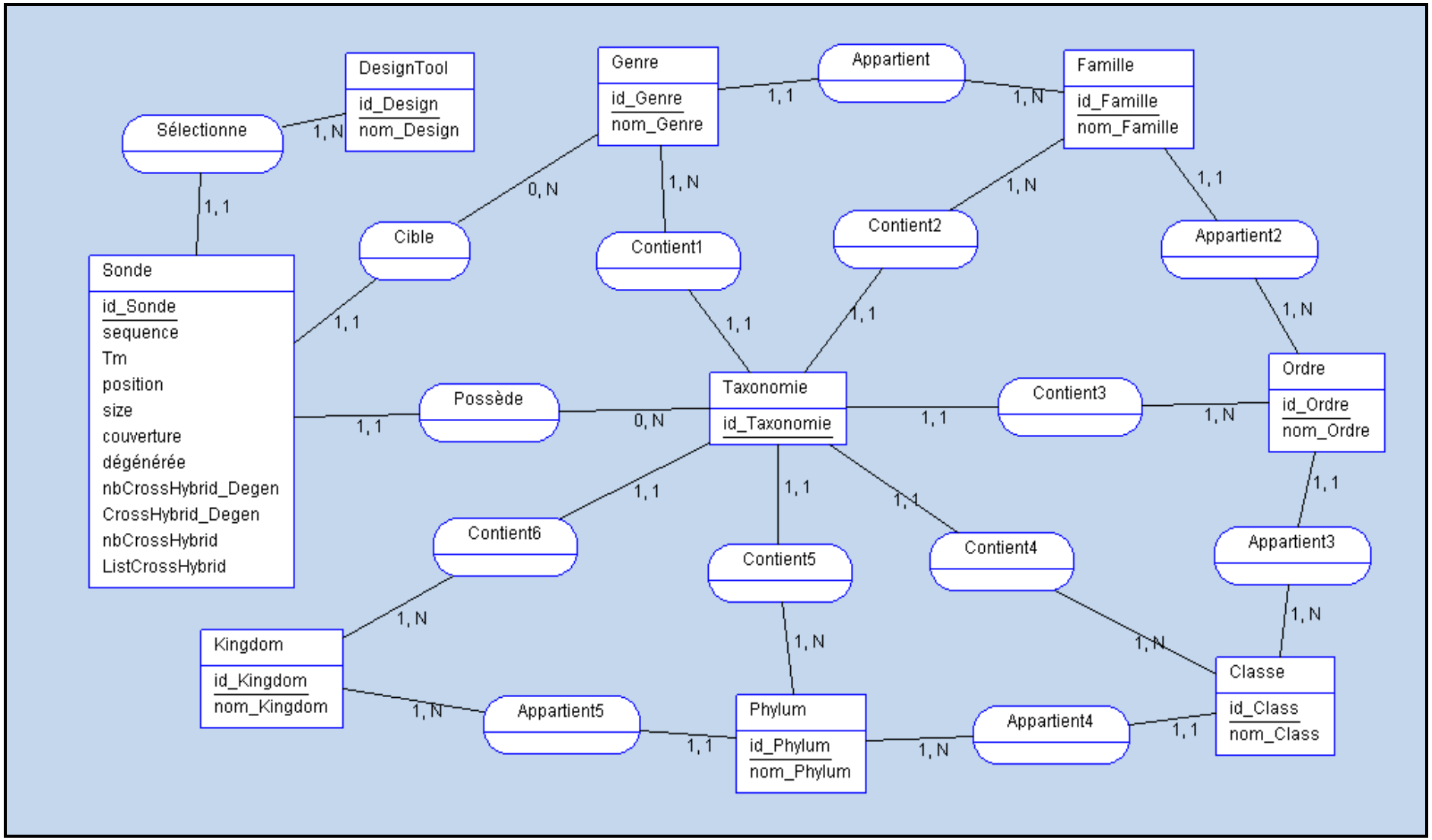

Figure 36. Modèle entité-association pour l'application web de PhylOPDb.

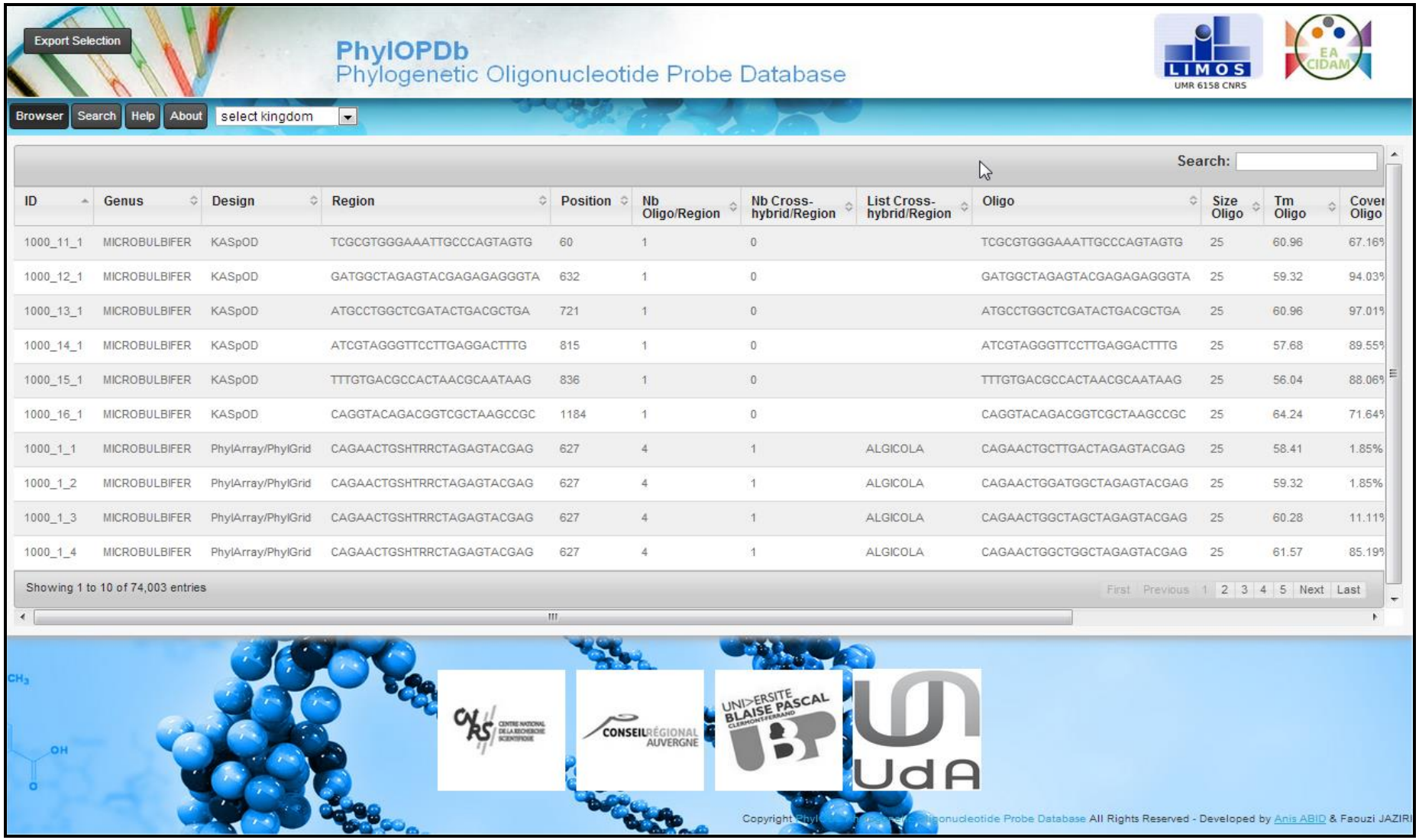

Figure 37. Interface web de PhylOPDb.

L'interface web de PhylOPDb permet une navigation hiérarchique du contenu de notre base de données de sondes, en se basant sur la taxinomie EMBL (figure 38). Lorsqu'un groupe taxinomique est sélectionné, seules les sondes oligonucléotidiques de ce groupe sont alors affichées et cette sélection de sondes peut être téléchargée. Un groupe taxinomique peut être un genre, une famille, un ordre, une classe, un phylum ou un règne. Le téléchargement des sélections de sondes affichées peut se faire à la fois en format CSV et FASTA. En outre, pour chaque sonde non dégénérée, les informations associées sont données dans le fichier CSV:

- ✓ **ID:** identifiant unique de la sonde (composé de trois parties séparées par le caractère « _ »: un identifiant numérique du genre ciblé, le numéro de la sonde dégénérée à partir de laquelle cette sonde est générée et le numéro de cette sonde).
- ✓ **Genus:** le nom du genre pour lequel cette sonde a été définie.
- ✓ **Design:** l'outil de design de sondes utilisé pour sélectionner cette sonde (KASpOD ou PhylGrid).
- ✓ **Region:** la séquence de la sonde dégénérée à partir de laquelle la sonde courante est générée.
- ✓ **Position:** la position de la sonde dégénérée sur la séquence consensus pour les sondes sélectionnées avec PhylGrid.
- ✓ **Nb Oligo/Region:** la dégénérescence de la région.
- ✓ **Nb Cross-hybrid/Region:** le nombre d'hybridations croisées potentielles de la sonde dégénérée avec des genres non ciblés.
- ✓ **List Cross-hybrid/Region:** la liste des genres qui peuvent avoir des hybridations croisées avec cette sonde dégénérée.
- ✓ **Oligo:** la séquence de la sonde non dégénérée extraites à partir de la sonde dégénérée (Identique à la séquence de la région si cette dernière est non dégénérée).
- ✓ **Size Oligo:** la taille de la sonde.
- ✓ **Tm Oligo:** la température de fusion de la sonde non dégénérée, calculée en utilisant la formule suivante: $Tm = 64.9 + 41 * (yG + zC - 16.4) / (wA + xT + yG + zC)$; dont w, x, y et z représentent respectivement les nombres des nucléotides A, T, G et C dans la séquence de la sonde.

- ✓ **Coverage Oligo:** le pourcentage de séquences du genre courant qui sont ciblées par cette sonde non dégénérée.
- ✓ **Nb Cross-hybrid/Oligo:** le nombre d'hybridations croisées potentielles de la sonde non dégénérée avec des genres non ciblés.
- ✓ **List Cross-hybrid/Oligo:** la liste des genres qui peuvent produire des hybridations croisées avec la sonde non dégénérée.

Les sondes peuvent également être obtenues et ensuite visualisées et/ou téléchargées en utilisant une recherche rapide par mots clé. Les mots clé peuvent représenter n'importe quel motif dans la base de données de sondes: une partie de la séquence d'une sonde, un nom de genre, le nom de l'outil de sélection de sondes utilisé (PhylGrid ou KASpOD), etc. Seules les sondes qui correspondent à ces mots clé sont alors affichées.

De plus, pour une recherche avancée permettant à l'utilisateur d'afficher et de télécharger une sélection de sondes qui répond à un ou plusieurs critères de choix, une page spécifique a été développée (figure 39). Cette page permet de filtrer les sondes de PhylOPDb en fonction des critères suivants: le nom du genre, l'outil de sélection de sondes utilisé, l'intervalle de température Tm, le nombre d'hybridations croisées et le pourcentage de couverture des séquences du genre ciblé. L'utilisateur peut utiliser un ou plusieurs de ces critères. La modification d'un critère implique une mise à jour instantanée de la sélection de sondes affichée, en prenant en compte seulement les critères choisis par l'utilisateur.

Notre site web contient une section « Help » contenant le manuel d'utilisation expliquant les différentes fonctionnalités de notre site à travers des exemples de navigation simples et commentés.

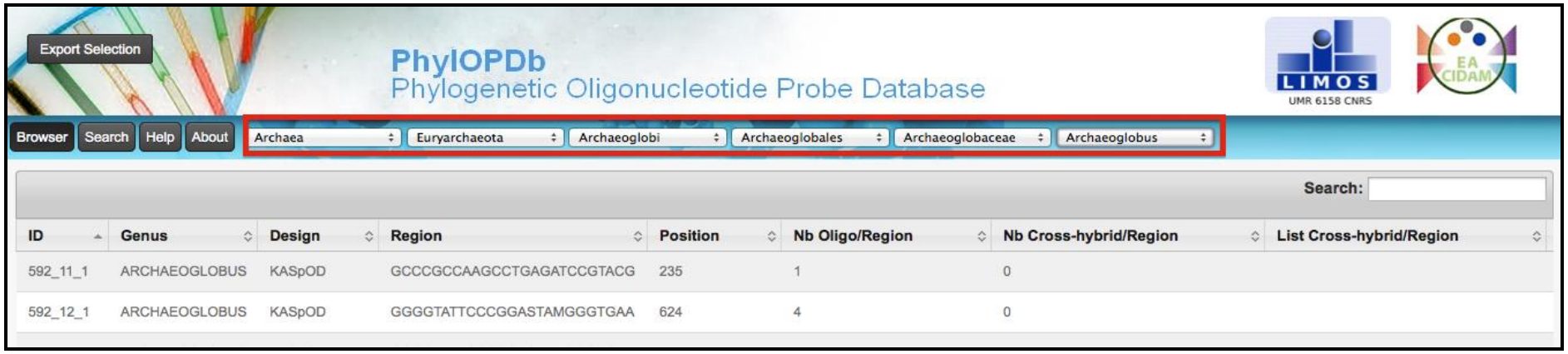

Figure 38. Exemple de navigation hiérarchique de PhylOPDb.

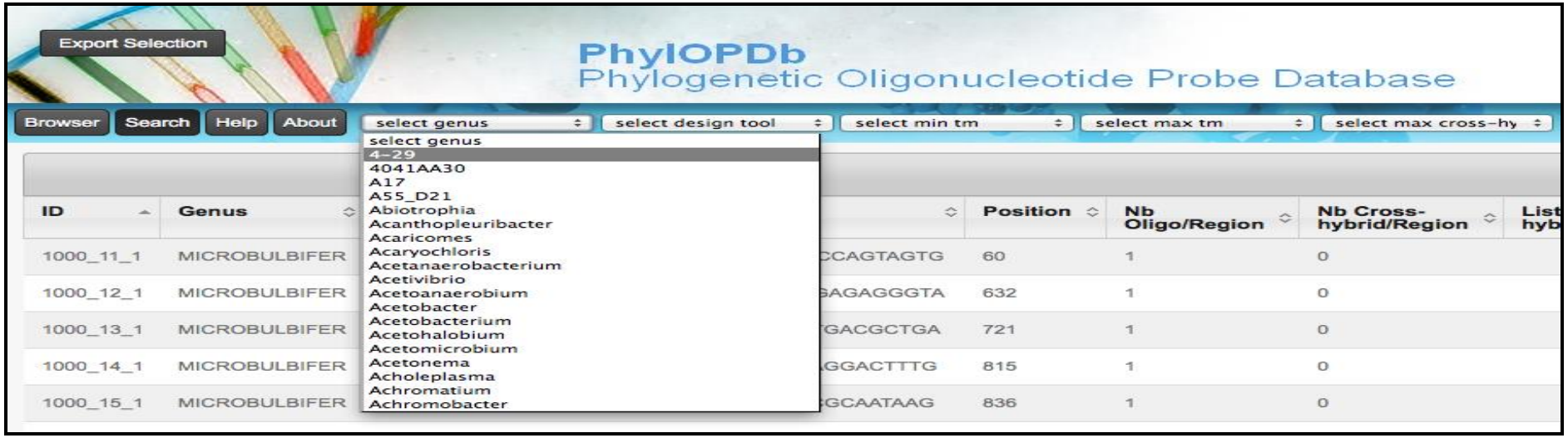

Figure 39. Recherche avancée de sondes.

3.3. Discussion

PhylOPDb fournit une interface web simple et conviviale pour consulter et télécharger librement une base de données complète de sondes ciblant le gène de l'ARNr 16S chez les procaryotes. Cette base de données est actuellement composée de 74 003 sondes de 25-mers régulières et exploratoires couvrant un total de 2178 genres procaryotes. Elle pourrait être utilisée pour construire une biopuce phylogénétique oligonucléotidique POA complète permettant ainsi le suivi de plus de deux milliers de genres microbiens en une seule expérience. PhylOPDb représente aujourd'hui la base de données la plus complète. En effet, comme nous l'avons montré, la détermination de sondes est particulièrement délicate pour ce biomarqueur et demande donc une expertise particulière. Signalons de plus, que seule cette base permet d'accéder à des sondes ciblant des espèces pour lesquelles aucune information de séquences n'est disponible.

Les sondes de la base de données PhylOPDb sont également bien adaptées à d'autres outils moléculaires que les biopuces à ADN (e.g. PCR, PCR quantitative, FISH). Ainsi, notre base de données est particulièrement utile pour des applications biologiques permettant de révéler des phylotypes particuliers et de déterminer les structures et les évolutions des communautés microbiennes de différents écosystèmes. Les sondes exploratoires renforcent la caractérisation de nouveaux taxons. Les sondes de la base de données sont également particulièrement bien adaptées pour les nouvelles approches innovantes de capture de séquences développées en écologie microbienne (Denonfoux et al., 2013).

Du fait, des connaissances limitées de la thermodynamique de l'hybridation des acides nucléiques les approches moléculaires notamment les biopuces à ADN utilisant les sondes et amorces demeurent encore largement empiriques et nécessitent des validations biologiques. Il a été en effet démontré que les approches *in silico* pour prédire le comportement d'hybridation des sondes de biopuces à ADN sont limitées: la meilleure solution étant d'effectuer un test expérimental très étendu des sondes pour assurer une spécificité complète (Loy and Bodrossy, 2006). Par conséquent, nous avons préféré fournir un ensemble complet de sonde pour permettre aux utilisateurs de pouvoir bénéficier d'un jeu de sondes leur permettant de répondre aux questions biologiques posées. Néanmoins, afin de réduire la complexité du choix des sondes utilisées, les meilleures signatures dégénérées à partir desquelles les sondes et les amorces spécifiques ont été déduites sont

indiquées dans PhylOPDb. Nous recommandons de plus aux utilisateurs de PhylOPDb de combiner plusieurs outils (par exemple Blast, SILVA-TestProbe, RDP-ProbeMatch) afin de confirmer les résultats de spécificité des sondes qui seront utilisées. En effet, les taxinomies évoluent constamment et il n'existe pas actuellement de bases de données universelles pour ce marqueur phylogénétique. De plus, la base de données de référence, les algorithmes de détermination des sondes et les paramètres de sélection peuvent influencer fortement les niveaux de spécificité.

Nos travaux futurs seront dirigés vers la détermination de sondes phylogénétiques pour des écosystèmes spécifiques. En effet, les tests de spécificité ne sont généralement pas réalisés sur le sous-ensemble approprié de séquences, principalement en raison de l'absence de bases de données spécialisées pour l'écologie microbienne. Selon l'environnement étudié, il serait plus pertinent d'effectuer ces tests de spécificité contre des bases de données de séquences réduites contenant uniquement les séquences pouvant être retrouvées dans les écosystèmes étudiés (e.g. sols, systèmes aquatiques, microbiotes intestinaux). Ainsi, des sondes encore plus spécifiques dédiées à l'exploration d'écosystèmes particuliers pourraient prochainement être accessibles via PhylOPDb.

D'autre part, les sondes fournies dans PhylOPDb ont été sélectionnées au niveau du genre, mais il serait également intéressant pour certains usages, comme les analyses utilisant la PCR, de définir des sondes ciblant d'autres niveaux phylogénétiques.

La collection de sonde actuelle de PhylOPDb pourra aussi être étendue pour inclure des sondes ciblant les séquences de l'ARNr 18S et permettant d'étudier par exemple les espèces fongiques ou d'autres microorganismes eucaryotes. D'autres biomarqueurs phylogénétiques pourraient également être exploités pour la conception de sondes permettant de distinguer des organismes ne pouvant être différenciés par l'analyse du gène de la petite sous-unité d'ARNr. Cependant, il faut dans ce cas de figure avoir une richesse d'informations assurant une fiabilité de détermination.

4. Conclusion

Dans ce chapitre nous avons présenté un nouveau programme pour la sélection de sondes oligonucléotidiques régulières et exploratoires pour biopuces à ADN phylogénétiques, sur grilles de calcul (avec gestion des resoumissions de jobs). Nous avons proposé une parallélisation efficace qui permet de distribuer la charge de calcul de sélection de sondes, équitablement, sur tous les cœurs utilisés. Nous avons testé et prouvé

la performance de notre programme sur des données biologiques réelles, en soumettant les calculs sur la grille de calcul européenne EGI (speedup de 106x avec 600 jobs, malgré la latence liée à l'attente et à la soumission des jobs).

Nous avons également proposé une nouvelle approche pour la construction des bases de données de séquences ciblant les gènes codant la petite sous unité de l'ARN ribosomique. Cette approche est bien adaptée aux bases de données utilisées pour la sélection de sondes pour biopuces phylogénétiques. Nous avons utilisé cette approche pour créer une base de données de séquences de l'ARNr 16S au niveau du genre. Cette base contient plus de 66 000 séquences pour un total de 2 069 genres procaryotes.

L'utilisation de PhylGrid2.0 et KASpOD (Parisot et al., 2012) a ensuite permis de produire la base de données, à ce jour, la plus exhaustive de sondes ciblant le biomarqueur 16S. Elle est également la seule base de données actuellement disponible en accès libre sur le Web et contenant des sondes exploratoires ciblant le biomarqueur 16S.

Dans le chapitre suivant, nous présentons un autre outil de sélection de sondes exploratoires, développé pour remédier aux limites de PhylGrid2.0.

Chapitre 5: Sélection de sondes à grande échelle pour biopuces exploratoires en environnements parallèles par une approche d'Ingénierie Dirigée par les Modèles

1. Introduction

Dans ce chapitre, nous présentons un nouveau logiciel parallèle, appelé « MetaExploArrays », pour la sélection de sondes oligonucléotidiques (sensibles, spécifiques, exploratoires et isothermes) ciblant une séquence ou un groupe de séquences nucléiques. Nous utilisons une approche de méta-programmation et d'ingénierie dirigée par les modèles, afin de réduire la complexité et le temps de calcul de notre logiciel. MetaExploArrays génère automatiquement les codes sources nécessaires pour sélectionner les sondes sur un PC, un multiprocesseur, un cluster ou une grille de calcul.

2. MetaExploArrays: logiciel de sélection de sondes à grande échelle pour biopuces exploratoires

2.1. L'implémentation

Nous avons implémenté notre méthode dans un programme que nous avons appelé « MetaExploArrays ». MetaExploArrays a été développé en C++ sous Linux CentOS 5.4. Il utilise deux autres programmes: Blastn (Altschul et al., 1990) et UNAFold (Markham and Zuker, 2008). L'utilisateur doit d'abord spécifier l'architecture souhaitée (PC, multiprocesseur, cluster ou grille de calcul), le nombre de processeurs à utiliser et la taille des sondes oligonucléotidiques à sélectionner. MetaExploArrays génère alors, en fonction de ces paramètres, les codes sources nécessaires au traitement de cette tâche de sélection de sondes. Les codes sources sont ensuite compilés et la sélection de sondes est exécutée. Cette méta-programmation a été réalisée en utilisant une approche d'ingénierie dirigée par les modèles. Le modèle de notre système est décrit par le diagramme de classe UML de la figure 40. Toutes les classes de ce diagramme décrivent des modèles de programmation ou d'architecture matérielle. Les classes « Prog-1 Seq », « Prog-Group of Sequences », « Prog-Tm Calculation » et « Prog-Segmentation » héritent de la classe « Program » et correspondent à:

- ✓ Prog-1 Seq: classe de sélection de sondes pour une seule séquence ADN.
- ✓ Prog-Group of Sequences: classe de sélection de sondes pour un groupe de séquences ADN.
- ✓ Prog-Tm Calculation: classe de calcul de la température de fusion Tm d'une sonde.

- ✓ Prog-Segmentation: classe de segmentation de l'ensemble des séquences dégénérées à traiter (partage de la charge de calcul).

En fonction de l'environnement de calcul choisi par l'utilisateur, la transformation, par génération de code, concerne les codes de segmentation et de sélection de sondes et les scripts shell nécessaires pour l'utilisation de l'architecture désirée. La génération de code effectuée correspond à une transformation de modèle au sens de l'IDM.

Deux types de fichier d'entrée sont acceptés par notre programme: un fichier FASTA contenant une séquence d'acide nucléique spécifique ou un fichier d'alignement, au format CLUSTAL, contenant le résultat d'un alignement multiple d'un groupe de séquences d'acides nucléiques. Pour la vérification de la spécificité des sondes, l'utilisateur peut fournir en entrée un fichier contenant une base de données de séquences d'acides nucléiques au format FASTA. Il peut aussi utiliser une des bases de données prédéfinies disponibles dans MetaExploArrays.

Les bases de données que nous mettons à la disponibilité des utilisateurs de MetaExploArays sont:

- ✓ Une base de données de haute qualité, en format FASTA, composée d'environ 66 000 séquences procaryotes du gène exprimant la petite sous unité de l'ARN ribosomique 16S. Ces séquences correspondent à 2 069 genres procaryotes (voir chapitre4 section 3 pour plus de détails). Cette base de données peut être utilisée pour sélectionner des sondes pour des biopuces phylogénétiques oligonucléotidiques POAs pour l'étude des communautés procaryotiqus.

- ✓ Une large base de données, en format FASTA, dédiée à l'étude des fonctions des communautés microbiennes. Elle est composée de toutes les séquences nucléotidiques codantes (CDS pour « Coding DNA Sequence ») annotées et retrouvés dans les divisions procaryotes, champignons et environnementales de la banque de données génomique EMBL. Cette base de données, appelée EnvExBase a été développée par l'équipe dans (Dugat-Bony et al., 2011) (voir chapitre2 section 2.3.1 pour plus de détails). Cette base de données peut donc être utilisée pour sélectionner des sondes pour biopuces fonctionnelles FGAs.

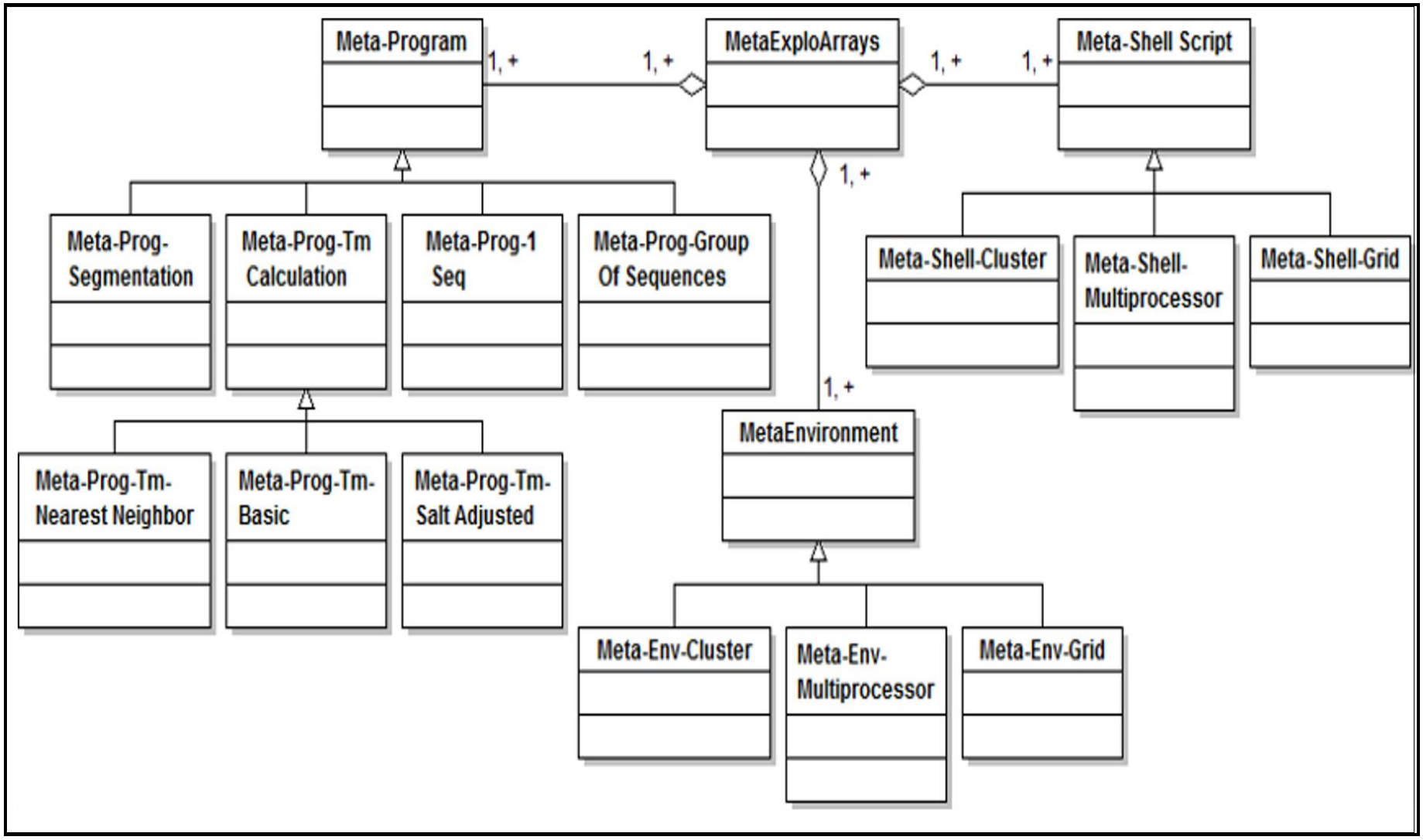

Figure 40. Méta modèle UML de l'algorithme de sélection de sondes proposé.

Avant de sélectionner des sondes oligonucléotidiques, l'utilisateur doit définir les valeurs des paramètres suivants:

- ✓ l: la longueur des oligonucléotides désirés,
- ✓ C: le nombre maximal autorisé d'hybridations croisées potentielles pour garder une sonde oligonucléotidique,
- ✓ s: le seuil de spécificité pour considérer une hybridation croisée potentielle,
- ✓ t: la Tm (température de fusion) minimale pour retenir une sonde oligonuclotidique,
- ✓ T: la Tm maximale pour retenir une sonde oligonucléotidique,
- ✓ M: la méthode utilisée pour calculer la température de fusion Tm des oligonucléotides. Notre algorithme permet d'utiliser l'une des méthodes suivantes: méthode basique (équation 1), méthode d'ajustement (équation 2) ou méthode thermodynamique du plus proche voisin (équation 3). MetaExploArrays génère automatiquement le code source de la méthode choisie en fonction du choix de l'utilisateur et de la longueur des oligonucléotides.

équation 1: $$T_m = 64.9 + 41 \times \left(\frac{yG + zC - 16.4}{wA + xT + yG + zC}\right)$$

équation 2: $$T_m = 79.8 + 18.5 * log10([Na^+]) + \left(58.4 * \frac{yG + zC}{wA + xT + yG + zC}\right) + \left(11.8 * \left(\frac{yG + zC}{wA + xT + yG + zC}\right)^2\right) - \left(\frac{820}{wA + xT + yG + zC}\right)$$

équation 3: $$T_m = \frac{\Delta H - 3.4 \frac{kcal}{°K\ mole}}{\Delta S + R.\ln\left(\frac{1}{C}\right)} + 16.6\ log_{10}([Na^+]) - 273.15$$

w, x, y et z sont respectivement les nombres des nucléotides A, T, G et C dans l'oligonucléotide testé; ΔH: la somme des enthalpies libres des couples de bases voisines (cal.mol^{-1}); ΔS: la somme des entropies des couples de bases voisines (cal.K^{-1}.mol^{-1}); R: la constante des gaz parfaits (1.987 cal.K^{-1}.mol^{-1}); C: la concentration moléculaire de l'oligonucléotide (mol.L^{-1}); [Na+]: la concentration en sels monovalents (mol.L^{-1}).

2.2. Sélection de sondes oligonucléotidiques pour une seule séquence nucléotidique

Plusieurs étapes sont nécessaires pour sélectionner des sondes à partir d'une seule séquence d'acide nucléique. Tout d'abord, la séquence d'entrée est lue pour extraire toutes les sondes possibles, en utilisant une fenêtre mobile de longueur égale à « *l* » (« *l* » est la longueur des sondes oligonucléotidiques fixée par l'utilisateur): On prend le premier oligonucléotide de taille « *l* » qui commence au premier nucléotide de la séquence traitée, ensuite, on avance par pas de « *1* » pour extraire les différentes sondes possibles.

Les sondes obtenues sont ensuite examinées pour éliminer celles qui sont estimées mauvaises selon l'ensemble de critères que nous donnons ci-dessous. De fait, nous ne gardons que les sondes qui répondent aux conditions suivantes:

- ✓ le pourcentage de nucléotides G+C est compris entre 40% et 60% de la longueur totale de l'oligonucléotide.
- ✓ l'oligonucléotide ne contient pas d'homopolymère (séquence composée du même nucléotide) de plus de 5 nucléotides.
- ✓ la température de fusion: t < = Tm < = T.

Si un oligonucléotide répond à ces critères, il est testé pour vérifier la présence de structures secondaires. Le programme UNAFold (Markham et Zuker, 2008) est utilisé pour calculer les énergies libres minimales de toutes les structures secondaires possibles (L'énergie libre ΔG mesure la stabilité de la réaction chimique à une température Tm et à une pression constante.) On utilise une concentration en *Na+* de 0,5 M et à une température égale à la moyenne de la plage de Tm définie par l'utilisateur (température = *(t + T) / 2*). Si, à cette température, un oligonucléotide contient une structure secondaire avec une énergie libre ΔG négative, il est alors rejeté.

La dernière étape est le test de spécificité des sondes. Le programme Blastn (Altschul et al., 1990) est utilisé pour vérifier la spécificité de toutes les sondes potentielles, contre la base de données de séquences choisie par l'utilisateur au début du programme. L'objectif est de déterminer toutes les hybridations croisées potentielles avec d'autres séquences différentes de celle ciblée. L'outil Blastn est exécuté avec les paramètres suivants:

- ✓ la taille du mot: « *-W 7* »,
- ✓ suppression du filtre des régions de faible complexité: « *–F F* »,

- ✓ l'e-value: « *-e 100* »,
- ✓ le nombre de résultats décrits sur une seule ligne (donne un aperçu rapide des résultats): « *-v 1* »,
- ✓ le nombre total d'alignements (résultats) à afficher: « *-b 1000* »,
- ✓ « *-S 1* » (indique quel brin nucléotidique à utiliser dans la recherche: brin d'entrée et/ou l'autre brin. Ici 1 pour le brin d'entrée).

Le fichier résultat fourni par le programme Blastn est ensuite analysé pour calculer, pour chaque oligonucléotide, le nombre d'hybridations croisées potentielles avec un seuil de spécificité supérieur à *S* (*S* est le seuil de spécificité pour considérer une hybridation croisée, ce seuil est un paramètre saisi par l'utilisateur). Si le nombre d'hybridations croisées est inférieur ou égal au nombre maximal fixé par l'utilisateur, l'oligonucléotide sera alors enregistré dans le fichier final de résultats avec les informations suivantes:

- ✓ la séquence de l'oligonucléotide,
- ✓ la position sur la séquence nucléotidique d'entrée,
- ✓ la liste d'hybridations croisées,
- ✓ Pour chaque hybridation croisée, le taux de similarité et les positions de mésappariements sont aussi donnés,

2.3. Sélection de sondes oligonucléotidiques pour un groupe de séquences nucléotidiques

Pour sélectionner des sondes pour un groupe de séquences d'acide nucléique qui représente un taxon donné, MetaExploArrays prend en entrée un fichier au format d'alignement Clustal. Ce fichier est le résultat de l'alignement multiple du groupe de séquences pour lequel nous allons sélectionner des sondes. Plusieurs étapes sont nécessaires (figure 41).

Premièrement, une séquence consensus est créée à partir du fichier d'alignement, en utilisant le code IUPAC des nucléotides dégénérés (Cornish-Bowden, 1985). L'objectif est non seulement d'obtenir une séquence commune qui représente tout le groupe de séquences nucléotidiques ciblé, mais aussi d'améliorer l'alignement et de corriger les erreurs potentielles de séquençage. En effet, pour chaque colonne de l'alignement, le nombre de nucléotides inconnus (« *N* » ou « - ») est calculé. Si ce nombre est supérieur à ½

NS (*NS* est le nombre total de séquences alignées), un gap « - » est inséré dans la séquence consensus dans cette position. Sinon, si ce nombre est supérieur à 0 mais inférieur à ½ *NS / 2*, toutes les bases non connues à cette position sont remplacées par la base dégénérée calculée en se basant sur seulement les bases non dégénérées connues à cette position.

Ensuite, notre algorithme lit la séquence consensus pour extraire toutes les sondes dégénérées possibles qui ne contiennent pas de gap (« - »), par l'incrémentation d'une fenêtre de longueur *« l »* (*« l »* est la longueur des oligonucléotides fixée par l'utilisateur) le long de la séquence consensus du groupe ciblé. Pour chaque sonde dégénérée, deux types de traitement sont réalisés: le premier est celui de la sélection des sondes régulières et le deuxième est celui de la sélection des sondes exploratoires.

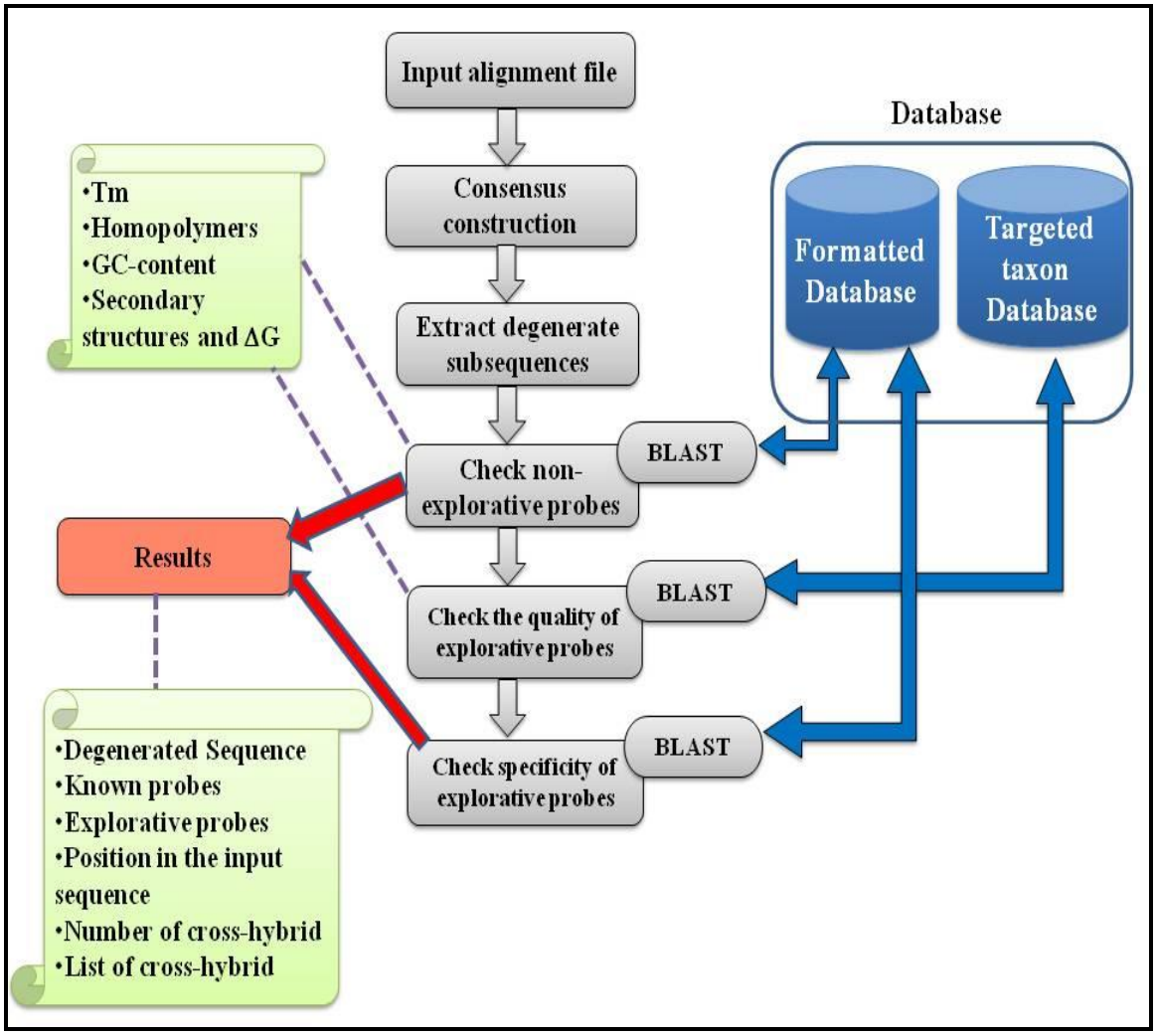

Figure 41. Etapes de sélection de sondes pour un groupe de séquences d'acides nucléiques.

2.3.1. Sélection des sondes régulières

Pour chaque sonde dégénérée *i* trouvée à la position *Pi* de la séquence consensus du groupe ciblé, MetaExploArrays calcule la valeur de la dégénérescence. Si cette valeur est supérieure à « *MaxDeg* », la sélection de sondes à partir de cette sonde dégénérée n'est alors pas autorisée. « *MaxDeg* » est la valeur maximale de dégénérescence pour retenir une sonde dégénérée. La valeur de la dégénérescence d'une sonde dégénérée varie d'un groupe à l'autre. Elle peut atteindre jusqu'à plusieurs millions d'oligonucléotides pour les groupes contenant une grande diversité de séquences. Dans le cas contraire, si la dégénérescence est inférieure à « *MaxDeg* », MetaExploArrays cherche toutes les sondes régulières potentielles correspondantes. Ces sondes sont extraites directement à partir de la position *Pi* de toutes les séquences nucléotidiques utilisées pour faire l'alignement du groupe ciblé, et non pas à partir de la sonde dégénérée. L'objectif est de considérer seulement les sondes existantes qui ciblent les séquences connues du groupe ciblé. Les sondes obtenues sont ensuite examinées pour ne garder que celles qui sont sensibles et isothermes (uniformes): la Tm, le pourcentage des bases G+C, l'absence d'homopolymères de plus de 5 nucléotides et l'absence de structures secondaires avec une énergie libre ΔG négative, sont ici vérifiés (comme décrit dans la section 2.2). Si l'une des sondes régulières extraites à la position *Pi* ne répond pas à tous ces critères, la sonde dégénérée *i* est alors supprimée et MetaExploArrays traite la prochaine sonde dégénérée. Dans le cas contraire, où toutes les sondes respectent les critères testés, MetaExploArrays passe alors à la vérification de la spécificité de ces sondes en utilisant le programme Blastn (Altschul et al., 1990).

L'étape du test de spécificité consiste à déterminer le nombre et la liste des groupes non ciblés, qui peuvent potentiellement s'hybrider avec les sondes régulières extraites. Tout d'abord, la base de données FASTA d'entrée, choisie par l'utilisateur, est préparée pour la recherche de similarité. En effet, nous commençons par supprimer toutes les séquences du groupe ciblé pour éviter le traitement des résultats inutiles et diminuer le temps de calcul. Les séquences des autres groupes sont ensuite dispersées et distribuées sur tout le fichier FASTA (figure 42), afin de détecter le maximum d'hybridations croisées avec des groupes différents. Cela permettra d'améliorer les résultats de la recherche de similarité et de diminuer le temps de calcul nécessaire pour la détection de toutes les hybridations croisées potentielles. Le programme Blastn est ensuite exécuté avec les paramètres suivants:

- ✓ « *-W 7* »,
- ✓ « *-F F* »,

- ✓ « *-e 1000* »,
- ✓ « *-v 1* »,
- ✓ « *-b 10000* »,
- ✓ « *-S 1* ».

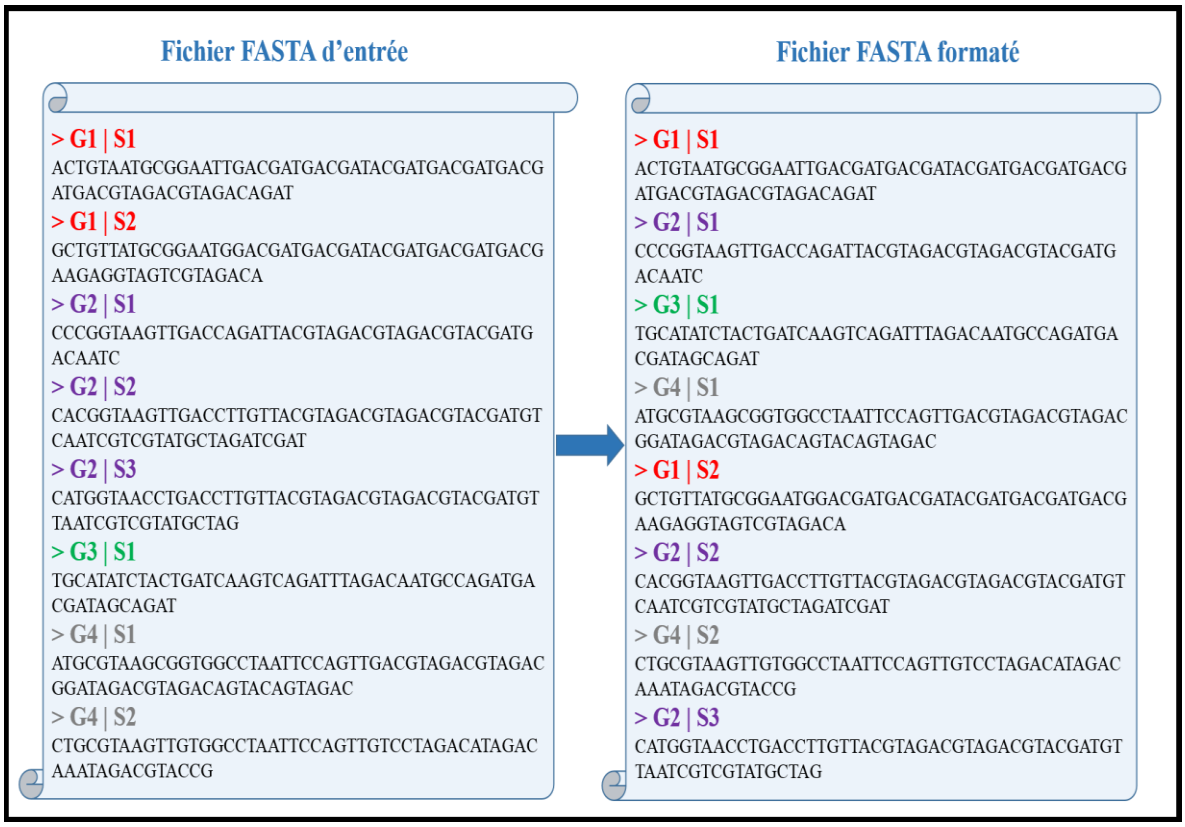

Figure 42. Exemple de transformation du fichier FASTA contenant la base de données de séquences pour le préparer à la recherche de similarité.

Enfin, MetaExploArrays analyse le fichier résultat fourni par Blastn pour déterminer les hybridations croisées potentielles de toutes les sondes testées avec les autres groupes non ciblés. Cette analyse se base sur le seuil de spécificité défini par l'utilisateur au début de l'exécution de notre programme. Le nombre et la liste des hybridations croisées non redondantes, de la sonde dégénérée actuelle, sont alors déterminés en additionnant les hybridations croisées de toutes les sondes régulières correspondantes. Si le nombre calculé est supérieur au nombre maximal autorisé d'hybridations croisées, MetaExploArrays supprime cette sonde dégénérée et vérifie la suivante. Dans le cas contraire, où elle

respecte le critère de spécificité, MetaExloArrays valide cette sonde dégénérée et l'enregistre dans le fichier résultat avec les informations suivantes:

- ✓ la séquence de la sonde dégénérée,
- ✓ la position de la sonde dégénérée sur la séquence consensus du groupe,
- ✓ le nombre et la liste complète des hybridations croisées potentielles de la sonde dégénérée,
- ✓ le nombre et la liste des sondes régulières correspondantes à cette sonde dégénérée,
- ✓ le nombre et la liste d'hybridations croisées potentielles pour chaque sonde régulière,
- ✓ pour chaque hybridation croisée potentielle, le niveau de similarité ainsi que le nombre de mésappariements et leurs positions sont décrits.

C'est exclusivement dans le cas où la sonde dégénérée courante répond à tous les critères de sélection des sondes régulières, conduisant à son archivage par MetaExploArrays, que notre programme exécute la prochaine étape qui consiste à déterminer les sondes exploratoires possibles pour cette sonde dégénérée.

2.3.2. Sélection des sondes exploratoires

Une fois l'étape de sélection des sondes régulières achevée, MetaExploArrays détermine, à partir de la sonde dégénérée courante, toutes les sondes exploratoires possibles qui ciblent des séquences appartenant potentiellement au groupe ciblé mais qui ne sont pas encore découvertes. D'abord, tous les oligonucléotides possibles sont générés à partir de la sonde dégénérée, en se basant sur le code IUPAC des nucléotides. Cette étape est réalisée par un script généré automatiquement par notre programme, en fonction de la longueur des oligonucléotides. Ce script utilise une technique de « pile » permettant de réduire la complexité et la quantité de RAM utilisée par cette tâche (Hill, 2006). Par exemple, pour une sonde composée de trois nucléotides dégénérés, le script généré correspond au pseudo-code donné par la figure 43. Ensuite, seulement les sondes exploratoires potentielles qui ne figurent donc pas dans la liste des sondes régulières sélectionnées sont traitées. Ces sondes sont testées pour vérifier l'absence d'homopolymères de plus de 5 nucléotides, le pourcentage des base G+C et la Tm. Seules les sondes qui respectent ces critères sont conservées, et les autres sont supprimées. Ici, MetaExploArrays ne vérifie pas l'absence de structures secondaires en même temps que

ces autres critères comme il le fait avec les sondes régulières, car cette tâche devient lourde en temps de calcul pour un grand nombre de sondes. Ce critère sera alors vérifié après la validation de toutes les autres étapes de sélection de sondes exploratoires et juste avant le test de spécificité. En effet, après chaque étape, l'ensemble des sondes retenues sera de plus en plus petit. La vérification des structures secondaires se fera alors sur un ensemble de sondes beaucoup plus réduit.

```
Begin
    For i = 1 to nb_nucléotide(1er nucléotide dégénéré)
        Push nucléotide(i) du 1er nucléotide dégénéré
        For j = 1 to nb_nucléotide(2ème nucléotide dégénéré)
            Push nucléotide(j) du 2ème nucléotide dégénéré
            For k = 1 to nb_nucléotide(3ème nucléotide dégénéré)
                Push nucléotide(k) du 3ème nucléotide dégénéré
                récupérer l'oligonucléotide obtenu de la pile
                Pop nucléotide(k) du 3ème nucléotide dégénéré
            End For
            Pop nucléotide(j) du 2ème nucléotide dégénéré
        End For
        Pop nucléotide(i) du 1er nucléotide dégénéré
    End For
End
```

Figure 43. Pseudo-code de génération de tous les oligonucléotides possibles à partir d'une sonde dégénérée (exemple pour une sonde de longueur 3 nucléotides dégénérés).

L'étape suivante de sélection de sondes consiste à ne retenir que des sondes exploratoires montrant une proximité avec le groupe ciblé afin d'éviter de conserver des sondes trop divergentes qui seront potentiellement non spécifiques. Dans ce cadre, le programme Blastn (Altschul et al., 1990) recherche les similarités entre les sondes retenues et une base de données contenant seulement les séquences d'acides nucléiques du groupe ciblé. L'objectif de cette étape est donc de mieux contrôler la qualité des sondes exploratoires. Ainsi, MetaExploArrays ne retient que les sondes qui ont une similarité avec au moins une des séquences du groupe ciblé avec un seul mésappariement maximum. Ce traitement permet, avec l'étape de la construction et de la correction de la séquence

consensus, de s'assurer que les sondes exploratoires choisies ciblent le bon groupe car elles sont issues de sa séquence consensus et sont aussi très proches de ses séquences connues.

Les sondes exploratoires retenues sont ensuite examinées pour vérifier si elles ne contiennent aucune structure secondaire avec une énergie libre négative. Seulement les sondes qui respectent ce critère seront examinées pour la recherche de similarités avec les séquences des autres groupes non ciblés. La spécificité de ces sondes est alors vérifiée contre le fichier de la base de données FASTA que nous avons préparé et formaté dans l'étape de sélection de sondes régulières. Pour valider le test de spécificité, une sonde exploratoire doit être très spécifique et ne doit pas présenter des hybridations croisées potentielles avec aucun autre groupe non ciblé. L'objectif ici est de s'assurer que les sondes exploratoires sélectionnées ciblent exclusivement le groupe taxonomique voulu. Cela permet d'améliorer la qualité de ces sondes et d'éviter les erreurs potentielles d'analyse et d'interprétation des résultats d'hybridation des biopuces à ADN.

À la suite de cette étape, seulement l'ensemble final de sondes exploratoires retenues après la validation de tous les critères, est enregistré avec les sondes régulières que nous avons déjà sélectionnées à partir de la sonde dégénérée courante. La prochaine sonde dégénérée est ensuite traitée.

Finalement, le résultat fourni à l'utilisateur consiste en un fichier qui contient toutes les sondes dégénérées valides du groupe ciblé ainsi que les sondes régulières et exploratoires correspondantes, avec toutes les informations nécessaires collectées sur ces sondes durant les différentes étapes du traitement.

3. Parallélisation de l'algorithme

L'expérimentation a montré que la sélection de sondes oligonucléotidiques pour une seule séquence d'acide nucléique, en utilisant MetaExploArrays, ne nécessite pas un temps de calcul long (quelques minutes: moins de 5 minutes pour une séquence ADN de 657 nucléotides en utilisant un seul cœur de calcul sur un processeur AMD Opteron à 2,3 GHz, voir résultats du tableau 10 dans la section 4.). MetaExploArrays traite alors ce type de tâche sur un simple PC. Toutefois, la sélection de sondes régulières et exploratoires pour un groupe de séquences d'acides nucléiques nécessite un temps de calcul beaucoup plus important pouvant atteindre jusqu'à plusieurs jours de calcul. MetaExploArrays permet alors l'exécution de ce type de sélection de sondes sur un multiprocesseur, un cluster ou une grille de calcul. Notre programme contient un générateur de scripts qui écrit

automatiquement les codes sources nécessaires pour paralléliser et exécuter une tâche de sélection de sondes sur l'architecture souhaitée. MetaExploArrays peut également réaliser une sélection de sondes pour plusieurs groupes de séquences simultanément.

Tout d'abord, l'utilisateur doit choisir l'architecture qu'il veut utiliser (un multiprocesseur, un cluster ou une grille de calcul), le nombre de processeurs à utiliser sur cette architecture, la liste des groupes de séquences ciblés et la méthode de calcul de la température de fusion Tm souhaitée. En fonction de ces choix, MetaExploArrays génère 2 types de codes sources:

- ✓ Les programmes C++ qui seront utilisés pour paralléliser et réaliser la tâche de sélection des sondes oligonucléotidiques pour tous les groupes de séquences ciblés.
- ✓ Les scripts Shell qui seront utilisés pour exécuter les programmes C++ sur l'architecture désirée, en utilisant le nombre de processeurs précisé par l'utilisateur.

Si l'utilisateur choisit d'utiliser une grille de calcul, notre logiciel génère (par transformation au sens IDM) des scripts supplémentaires pour contrôler les soumissions des jobs et leurs resoumissions en cas de problème.

Le choix final de l'architecture et du nombre de processeurs à utiliser pour une tâche de sélection de sondes revient toujours à l'utilisateur. Cependant, MetaExploArrays aide l'utilisateur à choisir la meilleure architecture disponible en fonction du nombre de groupes ciblés et de leurs tailles, ainsi que du nombre de processeurs disponibles. Par exemple, si l'utilisateur a décidé d'utiliser un multiprocesseur avec seulement deux cœurs de calcul, pour une tâche de sélection de sondes qui nécessite plusieurs jours de calcul (le temps de calcul nécessaire est estimé au début par MetaExploArrays), notre programme lui recommande alors d'utiliser plus de processeurs sur un multiprocesseur ou un cluster. Il recommande également d'utiliser une grille de calcul si l'utilisateur ne dispose pas de suffisamment de processeurs disponibles. Dans ce cas, MetaExploArrays indique également à l'utilisateur le nombre de processeurs à utiliser pour exécuter cette tâche sur une grille de calcul avec des jobs d'environ 10 à 12 heures au maximum. Notre approche aide alors l'utilisateur dans sa prise de décision en lui indiquant, parmi tous les équipements de calcul disponibles, la meilleure solution pour exécuter sa tâche de sélection de sondes.

Une fois son choix validé, l'utilisateur doit fournir les différentes informations nécessaires à l'utilisation de l'architecture souhaitée, comme par exemple le nom d'utilisateur, le nom ou l'IP de la tête du cluster, l'organisation virtuelle « VO » et le serveur de stockage « SE » pour la grille, etc.

Pour distribuer la charge de calcul de la tâche de sélection de sondes souhaitée, MetaExploArrays traite d'abord chaque groupe de séquences séparément. La séquence consensus, construite à partir du fichier d'alignement de chaque groupe de séquences, est lue afin d'extraire toutes les sondes dégénérées possibles qui ne contiennent pas de gap (« - »). La dégénérescence de chaque sonde dégénérée est ensuite calculée. Si cette dégénérescence est inférieure à « *MaxDeg* », une valeur de poids est alors calculée pour cette sonde dégénérée sur la base de sa dégénérescence, le temps de calcul estimé nécessaire à son traitement et la valeur de « *MaxDeg* ». La sonde dégénérée et la valeur de son poids sont enregistrées dans un fichier.

Une fois cette étape réalisée pour tous les groupes ciblés, toutes les sondes dégénérées retenues sont collectées et mises dans le même fichier. Ce fichier est ensuite découpé en « *N* » sous-fichiers (« *N* » est le nombre de processeurs à utiliser défini par l'utilisateur), en fonction de la valeur du poids de chaque sonde dégénérée et de la somme de toutes les valeurs de poids. Les sous-fichiers ainsi créés ont pratiquement le même poids (figure 44). Les sondes dégénérées de chacun de ces sous-fichiers seront exécutées par un processeur. Ainsi, la parallélisation se fait à deux niveaux: intra- et inter-conception. Les résultats des sondes dégénérées peuvent ensuite être classés par groupe ciblé en se basant sur les identifiants uniques que notre programme attribue au départ à chaque sonde dégénérée traitée. Ici, le regroupement de toutes les sondes dégénérées dans le même fichier de départ, quel que soit le groupe de séquences à qui elles appartiennent, permet une meilleure répartition de la charge de calcul entre tous les processeurs utilisés. Il augmente la probabilité de créer des sous-fichiers avec des temps de calcul plus équilibrés. Supposons, par exemple, que nous désirons sélectionner des sondes pour deux groupes de séquences *G1* et *G2* en utilisant deux cœurs de calcul. Pour chacun de ces deux groupes, nous avons extrait, à partir de leurs séquences consensus, deux sondes dégénérées qui respectent les critères choisis par l'utilisateur. La première sonde dégénérée du groupe *G1* possède une dégénérescence de 1000 et la deuxième une dégénérescence de 800. La dégénérescence totale de *G1* est alors égale à 1800. Pour le groupe *G2*, la première sonde dégénérée a une dégénérescence de 300 et la deuxième une dégénérescence de 100. La

dégénérescence totale de *G2* est alors égale à 400. Si nous traitons chaque groupe séparément, nous obtiendrons alors deux jobs. Le premier sélectionne des sondes pour le groupe *G1* et possède donc une dégénérescence de 1800 (égale à la dégénérescence totale de *G1)* et le deuxième job sélectionne des sondes pour *G2* et possède donc une dégénérescence de 400 (égale à la dégénérescence totale de *G2).* Ici, la charge de calcul du premier job est alors égale à 4,5 fois celle du deuxième job. Cependant, si nous regroupant toutes les sondes dégénérées issues des deux groupes *G1* et *G2*, et nous partageons ensuite le traitement de ces sondes sur deux jobs, nous obtiendrons alors deux jobs égaux (en termes de charge de calcul) dont chacun possède une dégénérescence de 1100. Dans ce cas, la charge de calcul est équitablement répartie sur les deux cœurs de calcul utilisés.

L'étape suivante consiste à distribuer le calcul sur tous les processeurs utilisés, de manière à traiter chaque sous-fichier de sondes dégénérées sur un seul processeur. Toutes les parties du calcul sont ensuite réalisées pour sélectionner des sondes régulières et exploratoires pour toutes les sondes dégénérées quel que soit le groupe à qui elles appartiennent (la sélection de sondes se fait en utilisant la procédure décrite dans la section 2,3 de ce chapitre). Enfin, quand le calcul est terminé, tous les fichiers résultats obtenus sont analysés pour extraire toutes les sondes sélectionnées et les regrouper par groupe ciblé. Un fichier résultat est alors créé pour chaque groupe.

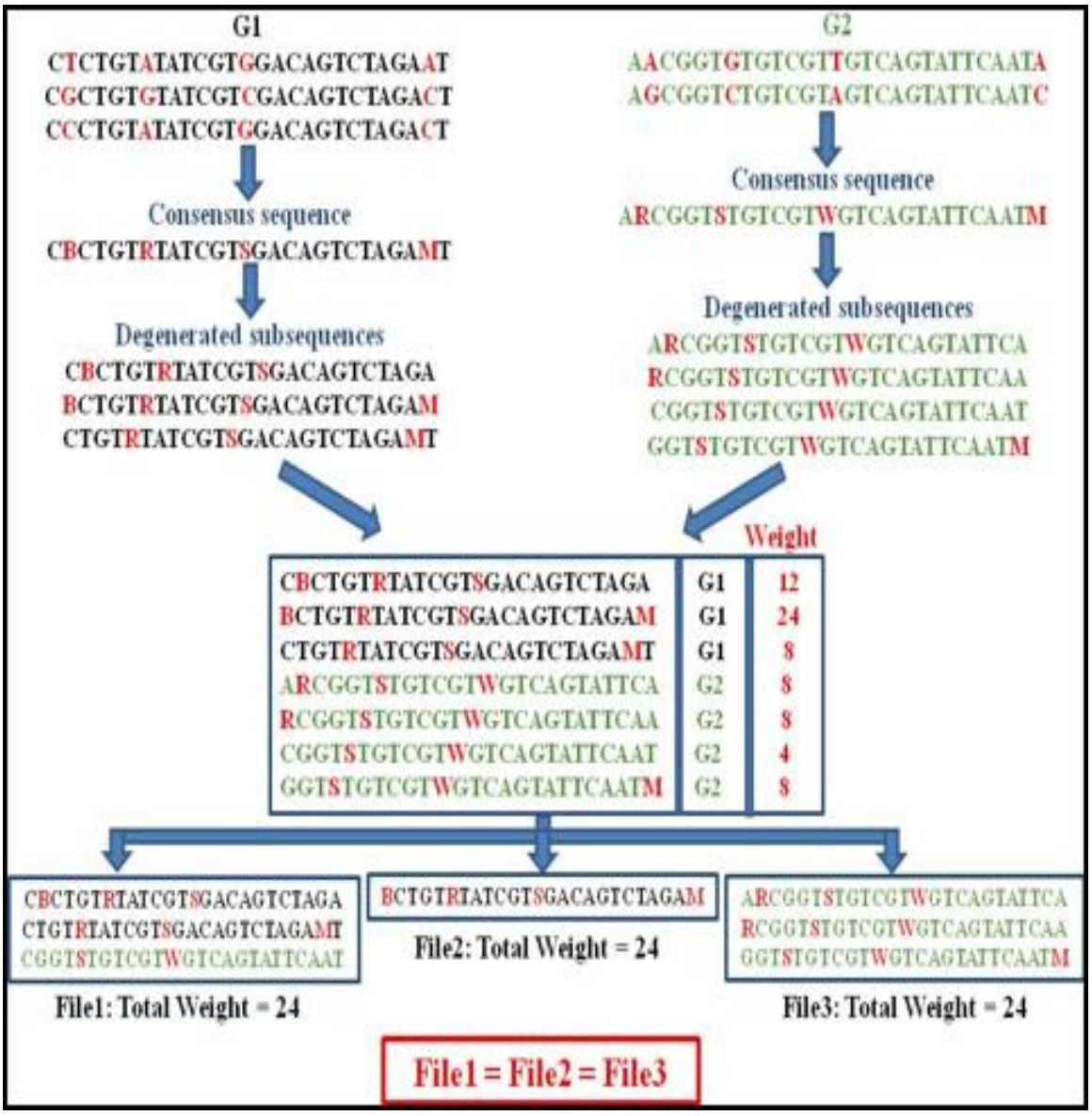

Figure 44. Exemple de parallélisation pour sélectionner des sondes ciblant deux groupes de séquences d'acides nucléiques, en utilisant 3 cœurs de calcul.

4. Résultats

MetaExploArrays utilise une méta-programmation basée sur une approche d'ingénierie dirigée par les modèles pour la génération (par transformation) des codes sources nécessaires à la sélection de sondes sur des architectures de calcul différentes: PCs, multiprocesseurs, clusters ou grilles de calcul. Cette approche nous a également permis d'améliorer l'étape de génération de tous les oligonucléotides possibles à partir d'une sonde dégénérée, en utilisant le code IUPAC des nucléotides. Cette étape est basée sur une pile capable d'amortir une partie de la nature exponentielle du problème en ce qui concerne

l'utilisation mémoire et le nombre de lectures écritures nécessaires à l'algorithme (Hill, 2006). La profondeur de la pile et le nombre de boucles imbriquées dépendent de la longueur de la séquence dégénérée à traiter. Ici, l'utilisation de la pile réduit le nombre d'écriture dans la mémoire et la quantité de RAM utilisée à cette étape. Cette quantité est précisément limitée à la taille de l'oligonucléotide, plus un certain nombre d'indexes de boucles. Pour vérifier le gain obtenu en utilisant cette technique, nous avons réalisé une comparaison entre MetaExploArrays et le logiciel PhylArray (Militon et al., 2007) qui utilise lui aussi une étape de génération de tous les oligonucléotides à partir d'une séquence nucléotidique dégénérée. Nous avons exécuté les tests de comparaison sur un processeur AMD Opteron à 2,3 GHz sous Linux. Les résultats sont présentés dans le tableau 8. Le deuxième exemple illustré dans le tableau 8 traite une séquence dégénérée d'une longueur égale à 64 bases pour extraire tous les oligonucléotides possibles. Les résultats de ces tests montrent clairement que notre programme est beaucoup plus efficace que PhylArray. Il réduit considérablement le temps de calcul et la quantité de mémoire vive consommée. En effet, même pour ce type de problème de petite taille, MetaExploArrays prend 1 minute et 43 secondes et seulement 11 Ko de RAM pour traiter cet exemple, tandis que PhylArray prend plus de 21 minutes, et consomme une très grande quantité de RAM qui dépasse 13 Go. Ici, le speedup de notre logiciel, par rapport à PhylArray, est d'environ 13 fois. De plus, la quantité de RAM consommée par notre logiciel est d'environ 1,2 10^6 fois plus petite que celle consommée par PhylGrid. Avec l'approche proposée dans MetaExploArrays, le gain en espace ira croissant par rapport à l'ancienne approche PhylArray. Plus la dégénérescence de la séquence consensus du groupe traité est élevée, plus le gain en RAM réalisé grâce à notre approche est important. Cela nous permet de traiter des séquences possédant une très grande dégénérescence que PhylArray ne peut pas traiter à cause de la quantité de RAM nécessaire.

Toutes les sondes régulières et exploratoires sélectionnées avec notre programme répondent à un nombre important de critères: pourcentage de G+C, Tm, homopolymères, structures secondaires et spécificité. Dans le tableau 9, nous présentons une comparaison entre les fonctionnalités de MetaExploArrays, celles de notre deuxième logiciel PhylGrid2.0, PhylArray (Militon et al., 2007) et deux autres logiciels de sélection de sondes populaires: OligoArray (Rouillard et al., 2003) et ORMA (Severgnini et al., 2009).

Tableau 8: Comparaison des méthodes de génération d'oligonucléotides à partir d'une séquence consensus, dans MetaExploArrays et PhylArray (Militon et al., 2007).

Séquence consensus	Taille sondes	Nb sondes dégénérées	Nb sondes générées	Temps de calcul (s)		RAM	
				PhylArray	MetaExploArrays	PhylArray	MetaExploArrays
ANTCRSGNBBCNANKTANNCBGATKBCNGC	25	6	21676032	40	3 **(13x)**	0,9 Go	11 Ko **(gain: 8 5792 fois)**
NTKCNRARANKSTNCNANTCNNTBANGSNBRANTKCNRARNNCSTNGNANACNNTBANGSTBNC	25	40	838860800	1288	103 **(≃13x)**	13,2 Go	11 Ko **(gain: 1,2 106 fois)**

Tableau 9: Comparaison de MetaExploArrays avec quatre autres logiciels de sélection de sondes.

Software	Critères de sélection					Sondes dégénérées	Sondes exploratoires	Parallélisation
	Spécificité	Tm	% G+C	faible complexité	Structures secondaires			
PhylArray	Blast	Basique	non	Non	Non	oui	oui	cluster
PhylGrid2.0	Blast	Ajustement	oui	Non	Non	oui	oui	grille
OligoArray	Blast et the-rmodynamique	Plus proches voisins	oui	Oui	oui	Non	non	multiprocesseur
ORMA	Blast	Ajustement	non	Oui	Non	oui	non	non
MetaExploArrays	Blast	-Basique -Ajustement -Plus proches voisins	oui	Oui	oui	oui	oui	-multiprocesseur -cluster -grille méta-programmation

Pour tester la performance de notre programme pour la sélection de sondes pour une seule séquence nucléotidique, nous avons exécuté trois tâches de détermination de sondes pour des séquences de tailles différentes appartenant à l'espèce *Eubacterium eligens*, en utilisant les mêmes paramètres d'entrée. Ce test est réalisé sur un seul cœur de calcul d'un processeur AMD Opteron à 2,3 GHz sous Linux. Les résultats des trois exemples sont présentés dans le tableau 10. Ces résultats montrent que le temps de calcul nécessaire pour la sélection de sondes pour une seule séquence est de l'ordre de quelques minutes (moins de 5 minutes pour une séquence de 657 nucléotides, environ 3 minutes pour une séquence de 447 nucléotides et moins de 2 minutes pour une séquence de 366 nucléotides).

Lorsqu'il s'agit d'une sélection de sondes pour plusieurs groupes de séquences simultanément, le temps de calcul nécessaire devient considérable. Pour remédier à ce problème de temps de calcul, nous avons proposé une méthode de parallélisation intra- et inter- groupes. Pour tester l'efficacité de notre méthode, nous avons réalisé une sélection de sondes pour le genre bactérien *Rhodovibrio*, en utilisant différents nombres de cœurs de calcul sur un multiprocesseur. Nous avons utilisé un SMP avec l'architecture suivante: 8 Quad Core AMD Opteron à 2,3 GHz sous Linux. Le temps nécessaire pour réaliser cette tâche sur un seul processeur est d'environ 80 minutes. Les résultats, illustrés dans le tableau 11 et la figure 45, prouve l'efficacité de notre méthode qui nous a permis de réaliser un speedup qui dépasse 8x en utilisant 10 processeurs et 16x en utilisant 20 processeurs.

Tableau 10: Résultats de sélection de sondes avec MetaExploArrays pour des séquences de l'espèce *Eubacterium eligens*.

Taille de la séquence	Paramètres					Temps
	Taille sondes	Méthode calcul Tm	Intervalle Tm	Spécificité	Hybridations croisées max	
657	25	Ajustement	35 – 70	0,92	10	4min22s
447	25	Ajustement	35 – 70	0,92	10	3min01s
366	25	Ajustement	35 – 70	0,92	10	1min49s

Tableau 11: Résultats de sélection de sondes avec MetaExploArrays pour le genre *Rhodovibrio* sur un multiprocesseur.

Nombre cœurs de calcul	Paramètres					Temps
	Taille sondes	Méthode calcul Tm	Intervalle Tm	Spécificité	Hybridations croisées max	
2	25	Plus proche voisins	35 – 70	0,92	20	42min21s
5	25	Plus proche voisins	35 – 70	0,92	20	17min54
10	25	Plus proche voisins	35 – 70	0,92	20	9min21
20	25	Plus proche voisins	35 – 70	0,92	20	4min55s

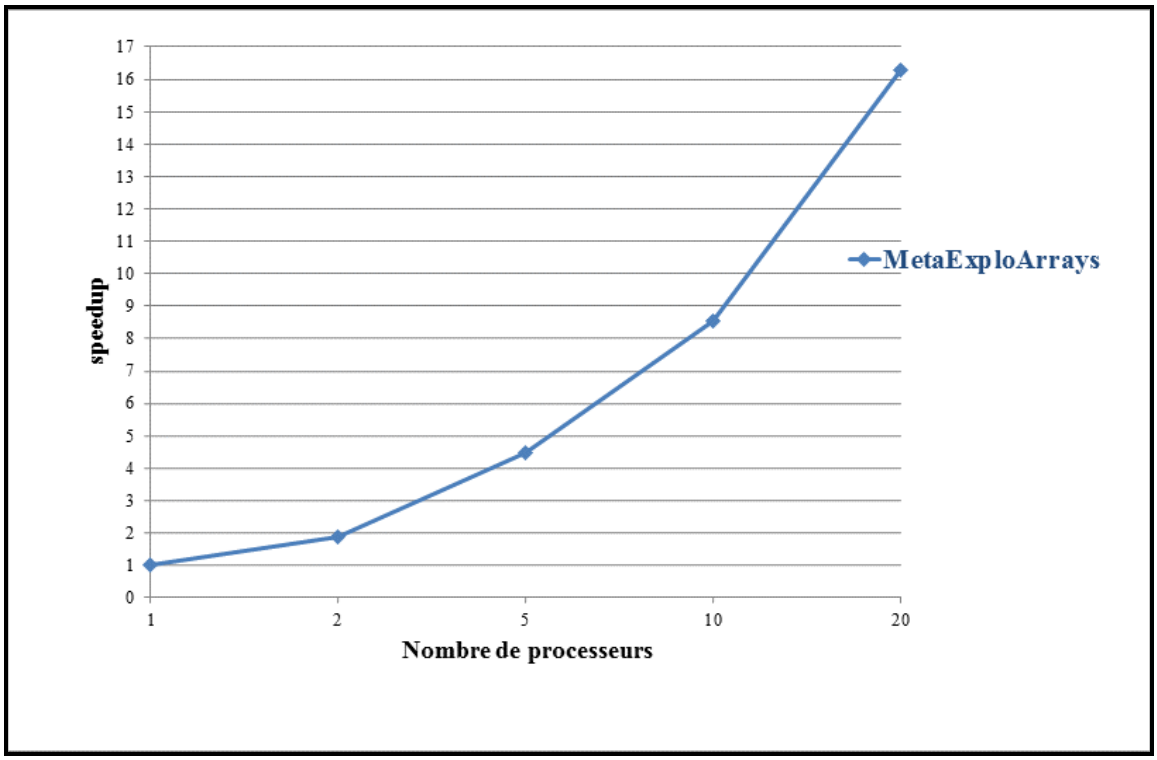

Figure 45. Speedup réalisé par MetaExploArrays lors de la sélection de sondes pour le genre *Rhodovibrio* sur un multiprocesseur.

Nous avons ensuite réalisé une autre sélection de sondes pour 20 genres procaryotes simultanément en utilisant 100 cœurs de calcul sur un cluster local. Ce cluster est composé de 22 nœuds bi-processeurs Quad Core AMD Opteron à 2,1 GHz sous Linux. Les résultats de ce test sont présentés dans le tableau 12. Ils montrent que la méthode de parallélisation utilisée dans MetaExploArrays a permis de réduire considérablement le temps de calcul nécessaire même pour des tâches de sélection de sondes pour plusieurs groupes de séquences de différentes tailles.

Toutefois, certaines tâches de sélection de sondes avec MetaExploArrays peuvent être très coûteuses en temps de calcul. C'est le cas de la sélection de sondes pour plusieurs grands groupes simultanément. Par exemple, pour développer une biopuce procaryote complète au niveau du genre, en utilisant la base de données de séquences 16S que nous avons présentée dans le chapitre 4, il faut sélectionner des sondes pour plus de 2000 genres. Cette tâche peut nécessiter plusieurs mois de calcul sur un seul processeur. Pour répondre aux besoins de calcul de ce type d'applications, MetaExploArrays offre la possibilité de sélectionner des sondes sur une grille de calcul. Il génère automatiquement les codes sources nécessaires pour la supervision des jobs soumis à la grille.

Tableau 12: Résultats de sélection de sondes avec MetaExploArrays pour 20 genres procaryotes, en utilisant 100 cœurs de calcul sur un cluster.

Groupes ciblés	Paramètres					Temps (heures)	
	Taille sondes	Méthode calcul Tm	Intervalle Tm	Spécificité	Hybridations croisées max	1 cœur de calcul	100 cœurs de calcul
CRONOBACTER; FLAVOBACTERIUM; MICROCOCCUS; STREPTOCOCCUS; TREPONEMA; CORYNEBACTERIUM; PSEUDOALTEROMONAS; PANTOEA; BRADYRHIZOBIUM; SPHINGOMONAS; HALOMONAS; NOCARDIA; VIBRIO; SHEWANELLA; BURKHOLDERIA; RHODOVIBRIO; HELICOBACTER; NOCARDIOIDES; MARINOBACTER; VIBRIO; GEOTHERMOBACTERIUM.	25	Ajustement	35 – 70	0,92	150	232,5 heures (9,68 jours)	6,5 heures

5. Discussion

Le nouveau logiciel de sélection de sondes oligonucléotidiques appelé MetaExploArrays que nous avons présenté permet de déterminer des sondes régulières pour une seule séquence d'acide nucléique ou des sondes régulières et exploratoires pour un groupe de séquences représentant un ou plusieurs organismes. MetaExploArrays prend en compte tous les critères habituels pour la sélection de sondes spécifiques, sensibles et isothermes. Pour chaque sonde, MetaExploArrays vérifie la spécificité, le nombre d'hybridations croisées potentielles, la température de fusion Tm, le pourcentage des bases G+C, la possibilité de formation de structures secondaires et l'absence d'homopolymères longs.

MetaExploArrays est l'un des rares outils qui offrent la possibilité de sélectionner des sondes exploratoires pour identifier de nouvelles espèces ou variants de gènes qui ne sont pas encore découverts. Par rapport à tous les autres logiciels (PhylArray (Militon et al., 2007), PhylGrid2.0 (Jaziri et al., 2014a); KASpOD (Parisot et al., 2012), HISpOD (Dugat-Bony et al., 2011) et MetabolicDesign (Terrat et al., 2011), MetaExploArrays a amélioré la stratégie de sélection de ce type de sondes. En effet, les sondes exploratoires sont strictement sélectionnées par MetaExploArrays sur la base d'une appartenance probable à un organisme ou un groupe d'organismes.

MetaExploArrays peut être utilisé pour développer des biopuces POAs (« Phylogenetic Oligonucleotide Arrays »), FGAs (« Functional Gene Arrays ») ou WGAs (« Whole Genome Arrays »), en fonction de la base de séquences utilisée pour la sélection de sondes. Les oligonucléotides sélectionnés par MetaExploArrays sont aussi adaptés à d'autres outils moléculaires qui utilisent des amorces ou des sondes: PCR, PCR quantitative, FISH et capture de gènes.

MetaExploArrays permet de sélectionner des sondes sur l'une des architectures de calcul suivantes: un PC, un multiprocesseur, un cluster ou une grille de calcul. Il contient un script permettant de guider l'utilisateur et de l'aider à choisir la meilleure architecture possible en fonction de la complexité de la tâche de sélection de sondes à réaliser et de la disponibilité des différentes architectures.

Notre logiciel comprend un générateur de programmes qui écrit automatiquement les codes sources nécessaires pour la sélection de sondes sur l'architecture souhaitée. Cette méta-programmation est basée sur une approche d'ingénierie générative dirigée par les

modèles. L'algorithme utilisé par le programme généré amortit la nature exponentielle du problème en ce qui concerne les temps de calcul (complexité réduite). Cet algorithme permet également de réduire la quantité de RAM consommée. Cela a permis par conséquent l'augmentation du nombre de sondes exploratoires pouvant être testées simultanément par notre logiciel. Ici, grâce à l'utilisation d'une technique de gestion de pile, les performances en temps de calcul de l'étape de génération de toutes les sondes possibles à partir d'une séquence consensus ont été augmenté de 13 fois par rapport à l'approche utilisée dans PhylArray (Militon et al., 2007), en utilisant un seul cœur sur un processeur AMD Opteron à 2,3 GHz (sans parallélisation).

Les performances globales de notre logiciel en temps de calcul ont été augmenté significativement (jusqu'à 16x en utilisant 20 cœurs de calcul sur un SMP avec 8 Quad Core AMD Opteron à 2,3 GHz pour la sélection de sondes pour le genre bactérien *Rhodovibrio*). Même pour des tâches de sélection de sondes pour plusieurs groupes de séquences de différentes tailles, les performances en temps de calcul ont été augmentées de manière significative (jusqu'à 36x pour la sélection de sondes pour 20 genres procaryotes, en utilisant 100 cœurs de calcul sur un cluster de 22 nœuds bi-processeurs Quad Core AMD Opteron à 2,1 GHz).

Avec l'évolution des architectures de calcul distribuées, les possibilités de MetaExploArrays peuvent être étendues par l'ajout de nouvelles architectures disponibles, pour permettre à l'utilisateur de sélectionner des sondes pour biopuces à ADN sur Cloud par exemple. Nous envisageons également, le développement d'une application web permettant aux communautés de bioinformaticiens et biologistes d'utiliser gratuitement notre logiciel pour la sélection d'ensembles de sondes adaptés à leurs propres travaux. Nous envisageons aussi d'utiliser MetaExploArrays pour développer des ensembles de sondes présélectionnées pour permettre aux biologistes de profiter de la qualité des sondes régulières et exploratoires sélectionnées à l'aide de notre logiciel, sans avoir besoin d'exécuter des tâches de sélection de sondes. Ces ensembles de sondes seront disponibles librement via l'interface web PhylOPDb que nous avons présentée dans le chapitre précédent.

6. Conclusion

Dans ce chapitre, nous avons proposé MetaExploArrays, un nouvel outil de sélection de sondes oligonucléotidiques régulières et exploratoires sur différentes architectures de

calcul intensif (un PC, un multiprocesseur, un cluster ou une grille de calcul). MetaExploArrays se base sur une approche de méta-programmation et d'ingénierie dirigée par les modèles pour la génération automatique des codes sources nécessaires pour la sélection de sondes.

Pour sélectionner des sondes pour les grands groupes de séquences, nous avons développé une méthode de parallélisation qui se base sur la fragmentation de la séquence consensus du groupe ciblé et sur la distribution équitable de toutes les sous-séquences obtenues, sur l'ensemble des processeurs disponibles. De plus, lorsque la tâche de sélection de sondes concerne plusieurs groupes simultanément, toutes les sous-séquences de ces groupes sont collectées et traitées ensemble. Ainsi, la parallélisation se fait sur deux niveaux: intra- et inter-design.

Nous avons développé MetaExploArrays afin d'améliorer et de faciliter la sélection des sondes. Cependant, les données issues de telles expériences nécessitent un traitement particulier et fastidieux. Dans ce cadre, nous présentons dans le chapitre suivant un nouvel outil d'analyse des résultats des biopuces à ADN développées pour MetaExploArrays ou pour d'autres outils de sélection de sondes pour des groupes de séquences tels que PhylArray (Militon et al., 2007), PhylGrid2.0 (Jaziri et al., 2014a) ou KASpOD (Parisot et al., 2012).

Chapitre 6: Nouvelle approche pour l'analyse des biopuces à ADN taxonomiques

1. Introduction

Dans ce chapitre nous présentons un nouvel algorithme pour l'analyse des résultats d'hybridation des biopuces à ADN utilisant des sondes développées à l'aide des logiciels de sélection de sondes que nous avons présentés dans les chapitres précédents: PhylGrid2.0 et MetaExploArrays. Cet outil étant généraliste, il est également adapté aux biopuces sur lesquelles sont positionnées des sondes déduites d'autres logiciels de sélection comme PhylArray (Militon et al., 2007) ou KASpOD (Parisot et al., 2012). Notre programme utilise les notions de la logique propositionnelle pour déterminer la composition d'un échantillon hybridé.

2. Définitions et généralités

Pour certains microorganismes, il est impossible de déterminer des sondes hautement spécifiques, c'est-à-dire n'engendrant aucune hybridation croisée potentielle avec un autre microorganisme. La solution est alors de se contenter des sondes les plus spécifiques en essayant de limiter le nombre d'hybridations croisées. Cependant, avec cette approche, l'analyse des résultats issus des biopuces contenant de telles sondes devient une tâche plus complexe et fastidieuse. En effet, il faut que l'outil d'analyse utilisé prenne en considération la spécificité des sondes afin d'éviter les erreurs potentielles d'interprétation des signaux d'intensités élevées causées par des hybridations croisées et non par des hybridations spécifiques. Ici, nous présentons un nouvel algorithme pour la détermination de la composition d'un échantillon hybridé sur une biopuce ADN.

2.1. Calcul propositionnel

Le calcul propositionnel peut être défini comme étant la logique des propositions.

Définition 1: *Une proposition (ou variable propositionnelle) est une variable booléenne qui prend ses valeurs de vérité parmi les constantes 0 (faux) et 1 (vrai), notées aussi ⊥ (faux) et ⊤ (vrai).*

Le langage de la logique propositionnelle est construit à partir d'un ensemble dénombrable de variables propositionnelles, des constantes 1 (vrai) et 0 (faux) (ou « ⊥ » et « ⊤ »), des parenthèses et des connecteurs booléens suivants:

- ✓ la négation: « ¬ » (non);

- ✓ la conjonction: « ∧ » (le “et” logique);
- ✓ la disjonction: « ∨ » (le “ou” logique);
- ✓ l’implication: « ⟶ » (si … alors);
- ✓ la double implication (ou l’équivalence): « ⟷ » (si et seulement si);

Les éléments de ce langage permettent de définir les formules propositionnelles.

Définition 2: *L’ensemble des formules d’un langage propositionnel est le plus petit ensemble de formules bien formées tels que:*

- ✓ *0 et 1 (respectivement ⊥ et ⊤) sont des formules;*
- ✓ *une variable propositionnelle est une formule;*
- ✓ *Si P et Q sont des formules alors:*
  - *¬P et ¬Q sont des formules;*
  - *P ∧ Q est une formule;*
  - *P ∨ Q est une formule;*
  - *P ⟶ Q est une formule;*
  - *P ⟷ Q est une formule.*

Dans une formule propositionnelle, une proposition (variable propositionnelle) P est appelée un « littéral positif » et sa négation ¬P un « littéral négatif ».

Définition 3: *On appelle littéral une proposition P ou sa négation ¬P.*

Dans un langage propositionnel, nous pouvons distinguer certaines formules propositionnelles particulières:

- ✓ Clause: une clause est une formule composée d’une disjonction finie de littéraux positifs ou négatifs. Exemple: « P ∨ ¬Q ∨ R ».
- ✓ Monôme: un monôme est une formule composée d’une conjonction finie de littéraux positifs ou négatifs. Exemple: « P ∧ ¬Q ∧ R ».

En logique propositionnelle, une connaissance peut s’exprimer sous différentes formes:

- ✓ Forme normale conjonctive « CNF »: une formule propositionnelle est sous forme normale conjonctive si et seulement si elle est une conjonction de clauses.
- ✓ Forme normale disjonctive « DNF »: une formule propositionnelle est sous forme normale disjonctive si et seulement si elle est une disjonction de monômes.
- ✓ Forme normale négative « NNF »: une formule propositionnelle est sous forme normale négative si et seulement si les seuls connecteurs logiques utilisés sont: la négation (¬), la conjonction (∧) et la disjonction (∨), et si la négation n'est utilisée que sur les variables propositionnelles.

La forme normale conjonctive est très utilisée pour la représentation de connaissances et pour la modélisation des problèmes (Gu et al, 1997; Sais, 2008). En effet, les connaissances sont typiquement acquises petit à petit et les problèmes difficiles à résoudre sont souvent les « over-constrained » (ceux qui contiennent des contraintes contradictoires). Le problème peut alors s'exprimer naturellement comme la conjonction de ces contraintes. Dans la suite, nous allons utiliser seulement la forme normale conjonctive que nous noterons formule CNF.

2.2. Problème de satisfaisabilité SAT

On appelle une interprétation d'une formule propositionnelle, toute affectation des valeurs {vrai, faux} à ses variables. Un modèle d'une formule est une interprétation qui satisfait cette formule.

Une formule CNF $\mathcal{F}$ est dite consistante (ou satisfaisable ou cohérente) s'il existe une interprétation $\mathcal{I}$ telle que pour toute clause ci de $\mathcal{F}$, $\mathcal{I}$ est un modèle de ci. On dit que $\mathcal{I}$ satisfait $\mathcal{F}$ et on note $\mathcal{I} \models \mathcal{F}$ (Exemple 1).

Une formule est dite inconsistante (ou insatisfaisable ou incohérente) si et seulement si elle n'admet aucun modèle (Exemple 2).

Exemple 1: Soit l'ensemble des littéraux {a,b,c} et une formule CNF $\mathcal{F}$ tel que $\mathcal{F}$ = (a ∨ b) ∧ (¬a ∨ c) ∧ (¬a ∨ ¬b). $\mathcal{I}$ = {a,¬b,c} est une interprétation telle que $\mathcal{I}$(a)=vrai, $\mathcal{I}$(b)=faux, $\mathcal{I}$(c)=vrai. $\mathcal{I}$ satisfait chaque clause de $\mathcal{F}$, donc $\mathcal{I}$ est un modèle de $\mathcal{F}$. On dit alors que $\mathcal{F}$ est satisfaisable.

Exemple 2: Soit l'ensemble des littéraux {a,b} et une formule CNF $\mathcal{F}$ tel que $\mathcal{F}$ = (a ⋁ ¬b) ⋀ (¬a ⋁ b) ⋀ (¬a ⋁ ¬b) ⋀ (a ⋁ b). Les interprétations possibles pour cette formule $\mathcal{F}$ sont:

- ✓ $\mathcal{I}_1$ = {a,b} ne satisfait pas la clause (¬a ⋁ ¬b) donc $\mathcal{I}_1$ n'est pas un modèle de $\mathcal{F}$.
- ✓ $\mathcal{I}_2$ = {¬a,b} ne satisfait pas la clause (a ⋁ ¬b) donc $\mathcal{I}_2$ n'est pas un modèle de $\mathcal{F}$.
- ✓ $\mathcal{I}_3$ = {a,¬b} ne satisfait pas la clause (¬a ⋁ b) donc $\mathcal{I}_3$ n'est pas un modèle de $\mathcal{F}$.
- ✓ $\mathcal{I}_4$ = {¬a,¬b} ne satisfait pas la clause (a ⋁ b) donc $\mathcal{I}_4$ n'est pas un modèle de $\mathcal{F}$.

Dans cet exemple, aucune interprétation possible ne satisfait toutes les clauses de $\mathcal{F}$, donc $\mathcal{F}$ est insatisfaisable.

Le problème SAT ou problème de satisfaisabilité booléenne (« boolean SATisfiability problem ») est un problème de décision qui consiste à déterminer si une forme normale conjonctive $\mathcal{F}$ est satisfaisable (admet ou non un modèle) (Sais, 2008). La formule $\mathcal{F}$ est ici appelée une instance SAT.

Le problème SAT a été très étudié en informatique, du fait de son importance aussi bien théorique que pratique (Mitchell et al., 1992). En fait, il occupe un rôle central en théorie de la complexité (Schaefer, 1978; Allender et al., 2009, Creignou et al., 2001): c'est le premier problème à avoir été montré NP-complet (« Non déterministe Polynomial ») par Cook en 1971 (Cook, 1971). Le problème SAT est au cœur de nombreuses applications en intelligence artificielle (Lange, 2005; Heljanko et al., 2012). En plus, nombreux problèmes provenant d'autres domaines peuvent être réduits au problème de satisfiabilité, comme par exemple la bioinformatique (Lynce and Marques-Silva, 2006; Manolios et al., 2007), l'ordonnancement (Kautz and Selman, 1998), la vérification des circuits électroniques (Biere et al., 1999), la vérification des logiciels (Jackson and Vaziri, 2000), la vérification des microprocesseurs (Velev and Bryant, 2001), le diagnostic (Smith et al., 2004), la cryptographie (Potlapally et al., 2007), etc. En effet, l'idée générale est de transformer automatiquement un problème en une instance SAT et d'utiliser un solveur SAT (un programme qui résout le problème SAT) pour le résoudre, au lieu d'avoir un algorithme de résolution spécifique à chaque problème.

Plusieurs algorithmes de résolution du problème SAT existent, par les plus utilisés récemment nous pouvons citer GSAT (Selman et al., 1992), Walksat (Selman et al., 1996), Chaff (Moskewicz et al., 2001), ASSAT (Lin and Zaho, 2004), MiniSat (Een and Sörensson, 2004, 2005), RSat (Pipatsrisawat and Darwiche, 2006), BerkMin (Goldberg and Novikov, 2007) et Z3 (De Moura en Bjoner 2008). Ces algorithmes sont basés sur différentes approches telles que la résolution (Robinson, 1965, Dechter and Rish, 1994), la programmation linéaire (Blair et al., 1986), le comptage du nombre de solutions (Iwama, 1989), la recherche locale (Selman et al., 1992), les algorithmes génétiques (Hao et al., 2002, Lardeux et al., 2006), etc.

Les algorithmes de résolution SAT peuvent être classés en deux types: les algorithmes complets et les algorithmes incomplets.

Les algorithmes incomplets (ou de recherche stochastique) n'explorent pas l'espace de recherche de manière extensible et systématique. Ils consistent à rechercher un modèle rapidement. La durée de ces algorithmes est alors bornée et raisonnable. Cependant, ces algorithmes ne prouvent pas l'insatisfaisabilité du problème SAT car ils ne sont pas certains d'avoir parcouru toutes les interprétations possibles. Plusieurs solveurs reposant sur une méthode incomplète existent tels que GSAT (Selman et al., 1992), Walksat (Selman et al., 1996), TSAT (Mazure et al., 1997), G^2WSAT (Li et al., 2007), TNM (Wei and Li, 2009), CCA2013 (Li and Fan, 2013), etc.

Les algorithmes complets sont capables de trouver si une solution est possibleou de vérifier l'inexistence d'une solution, quel que soit l'instance SAT traitée. Ces algorithmes consistent à parcourir tout l'espace de recherche. Toutefois, si la formule est satisfaisable, le solveur s'arrêtera avant de tester les autres interprétations possibles. Le nombre d'étapes nécessaire pour ce type d'algorithme est d'ordre exponentiel en fonction du nombre total de littéraux. Il est par conséquent difficile de résoudre, dans un temps raisonnable, certains problèmes de très grande taille. Plusieurs solveurs complets ont été proposés: zchaff (Zhang et al., 2001), MiniSat (Een and Sörensson, 2004, 2005), RSat (Pipatsrisawat and Darwiche, 2006), BerkMin (Goldberg and Novikov, 2007), etc. Dans ce travail, nous avons utilisé le solveur zchaff qui est un solveur populaire et performant basé sur l'algorithme Chaff (Moskewicz et al., 2001) qui est l'algorithme le plus cité dans le domaine des SAT. Son code source est ouvert et disponible gratuitement. Il a été sélectionné comme meilleure solveur SAT complet pour les instances des catégories « industrial » et « handmade » dans

la compétition internationale des solveurs SAT[1] « SAT 2002 » et le meilleure solveur complet dans la catégorie « industrial » dans la compétition « SAT 2004 ».

3. PhylInterpret: algorithme d'analyse des biopuces à ADN pour la détermination de la composition d'un échantillon

3.1. Implémentation

Nous avons implémenté notre approche dans un programme que nous avons appelé « PhylInterpret ». PhylInterpret a été développé en C++ sous Linux CentOS 5.4. Il utilise deux autres programmes: Blastn (Altschul et al., 1990) et zchaff (Zhang et al., 2001).

L'utilisateur doit d'abord fournir un fichier qui contient toutes les sondes utilisées sur la biopuce ainsi que leurs séquences nucléotidiques, le groupe à qui elles appartiennent et la sonde dégénérée (ou la région) de laquelle elles sont issues. Chaque ligne contient les informations d'une sonde séparées par une tabulation. De plus, si le logiciel d'analyses des images des biopuces ADN (voir chapite1 section 4.4 pour plus de détail) permet de déterminer une valeur de bruit de fond associée avec chaque sonde, l'utilisateur doit fournir:

- ✓ Un fichier qui contient les intensités d'hybridation des sondes et les valeurs de bruit de fond correspondantes.

Sinon, si le résultat du logiciel d'extraction des données numériques à partir des images de biopuces ne contient pas de valeurs de bruit de fond, l'utilisateur doit fournir:

- ✓ Un fichier qui contient les intensités d'hybridation des sondes et leurs positions sur la biopuce.
- ✓ Un fichier qui contient les sondes de contrôle négatif utilisées ainsi que leurs intensités et leurs positions sur la biopuce.

L'utilisateur doit également fournir un fichier au format FASTA qui contient la base de données de séquences utilisée pour la sélection de sondes. Toutefois, dans le cas où il a utilisé PhylArray ou PhylGrid2.0 ou KASpOD ou MetaExploArrays, avec les bases de données par défaut de ces outils, il n'est pas nécessaire de redonner cette base qui est déjà disponible avec notre programme.

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

3.2. Approche Proposée

3.2.1. Prétraitement

Pour déterminer la composition d'un échantillon biologique hybridé sur une biopuce à ADN, nous prenons en considération la spécificité des sondes utilisées dans l'expérience d'hybridation. D'abord, toutes les sondes sont vérifiées pour déterminer, pour chacune, toutes les hybridations croisées potentielles avec des groupes de séquences non ciblés. Cette étape est effectuée à partir de la base de données de séquences utilisée lors de la tâche de sélection de sondes et le programme Blastn avec les paramètres suivants:

- ✓ « -W 7 »,
- ✓ « -F false »,
- ✓ « -e 1000 »,
- ✓ « -v 1 »,
- ✓ « -b 10000 »,
- ✓ « -S 1 ».

Le fichier résultat du programme Blastn est ensuite parcouru pour extraire toutes les similarités calculées et qui ont un seuil d'identité supérieur ou égal à « *seuil* » avec les sondes testées (*seuil* est le seuil de spécificité fixé par l'utilisateur pour considérer une hybridation croisée potentielle). Ces similarités sont ensuite regroupées par le nom du groupe afin de construire, pour chaque sonde, une liste non redondante des groupes avec lesquels elle représente des hybridations croisées potentielles.

Pour les grands formats de biopuces contenant jusqu'à des dizaines de milliers de sondes différentes, cette étape de test de spécificité peut nécessiter un temps de calcul très important. Pour cela, nous avons parallélisé cette étape pour l'exécuter sur un cluster de calcul. En effet, l'ensemble total des sondes utilisées sur la biopuce est partagé équitablement en « *N* » sous fichiers (*N* est le nombre de processeurs utilisés, sa valeur est donnée par l'utilisateur). Chaque sous fichier sera exécuté sur un cœur de calcul. La soumission et l'ordonnancement des jobs sont réalisés à l'aide de « PBS » (« Portable Batch System[2] »).

[2] http://www.pbsworks.com/

La deuxième étape de notre approche consiste à regrouper les valeurs de tous les réplicats de sondes utilisés sur la biopuce. Soit S l'ensemble de toutes les sondes de la biopuce. Pour chaque sonde 's', telle que 's' $\in S$, une valeur d'intensité d'hybridation médiane est calculée. Cette intensité est notée I_S.

Ensuite, l'étape suivante est celle de la détermination de la réponse (présence ou non dans l'échantillon hybridé) de chaque sonde. Deux cas sont possibles. Si l'utilisateur a fourni en entrée les valeurs des bruits de fond locaux relatives aux sondes, alors une valeur appelée « intensité minimale de réponse » et notée I_{Smin} est calculée pour chaque sonde. Cette valeur est dans ce cas égale à la valeur du bruit de fond du réplicat de cette sonde qui correspond à l'intensité d'hybridation médiane déjà sélectionnée. Une valeur *SNR* (« Signal to Noise Ratio ») est ensuite calculée comme suit:

$$\forall s \in S, SNR_S = \frac{I_S}{I_{Smin}}$$

Cependant, si l'utilisateur n'a pas fourni les valeurs des bruits de fond en entrée, un traitement supplémentaire est alors réalisé en utilisant les sondes de contrôle négatives de la biopuces. D'abord, pour chaque sonde, nous considérons seulement le réplicat qui a été choisi pour le calcul de l'intensité d'hybridation médiane. Ensuite, nous calculons pour ce réplicat, en fonction de ses positions x et y et de celles de toutes les sondes de contrôle de la biopuce, la distance maximale qui le sépare des sondes contrôles. On note cette distance d_{Smax}. Puis, nous nous basons sur cette valeur d_{Smax} ainsi que les positions de chaque sonde et de celles de toutes les sondes contrôles utilisées, pour calculer l'intensité minimale de réponse notée I_{Smin}, comme suit:

On définit par C l'ensemble de toutes les sondes de contrôle négatives utilisées sur la biopuce, et S l'ensemble de toutes les autres sondes de la même biopuce.

$\forall s \in S \; et \; \forall c \in C$, *on définit par* $d(s,c)$ *la distance entre s et c tel que*:

$$d(s,c) = \sqrt{(X_C - X_S)^2 + (Y_C - Y_S)^2}$$

où (X_C, Y_C) *les coordonnées de c et* (X_s, Y_s) *les coordonnées de s.*

$\forall s \in S et \forall c \in C$, *on définit par* P_{CS} *le poids de la sonde contrôle c relatif à la sonde s tel que*:

$$P_{CS} = f(d(s,c), d_{Smax}) = \frac{d(s,c)}{d_{Smax}}$$

Où d_{Smax} *est la distance entre s et la plus distante sonde de contrôle négative.*

$\forall s \in S et \forall c \in C$, *on définit par* I_{Smin} *la valeur minimale de réponse de s tel que*:

$$I_{Smin} = \frac{\sum_{c=0}^{n-1} (P_{CS} * I_C)}{n}$$

Où I_C *est l'intensité d'hybridation de c et n le nombre total des sondes de contrôle négatives de la biopuce.*

Enfin, nous calculons une valeur *SNR (« Signal to Noise Ratio »)* pour chaque sonde en utilisant la formule suivante:

$$\forall s \in \mathcal{S}, SNR_S = \frac{I_S}{I_{Smin}}$$

Pour déterminer si l'intensité d'hybridation d'une sonde '*s*' est suffisante pour décider que '*s*' s'est bien hybridée avec une séquence de l'échantillon traité, nous comparons sa valeur SNR_S avec un seuil de réponse T_{Sonde} dont la valeur est donnée par l'utilisateur. Si SNR_S est supérieure ou égale à T_{Sonde}, on dit que '*s*' a une réponse positive dans l'échantillon hybridé.

La quatrième étape de notre programme d'analyse des données de biopuces consiste à vérifier la réponse de chaque sonde dégénérée (ou région). Pour cela, nous regroupons toutes les sondes traitées dans les étapes précédentes, par sonde dégénérée. Si pour une sonde dégénérée donnée, au minimum une sonde qui lui appartient possède une réponse positive, alors on dit que cette sonde dégénérée a une réponse positive dans l'échantillon traité ou on dit encore qu'elle est positive. Sinon cette sonde dégénérée est dite négative.

La cinquième étape consiste à déterminer si un organisme ou ensemble d'organismes (selon le niveau de résolution des sondes) ciblé par la biopuce, est probablement présent ou non dans l'échantillon hybridé. Cela consiste à regrouper, par nom d'organisme(s), toutes les sondes dégénérées vérifiées dans l'étape 4. Ensuite, le pourcentage des sondes dégénérées positives par organisme(s) est calculé. Si ce pourcentage est supérieur ou égal au seuil de réponse $T_{organisme(s)}$ dont la valeur est fixée par l'utilisateur, on dit alors que ce

ou ces organismes sont probablement présents dans l'échantillon hybridé (ou groupe positif) et on le sauvegarde dans une liste préliminaire des organismes positifs. On note cette liste $\mathcal{L}_{G.}$

Lorsqu'un groupe de séquence est positif, deux cas sont possibles:

- ✓ Soit le ou les organismes sont réellement présents dans l'échantillon biologique hybridé,
- ✓ Soit ils ne sont pas présents dans l'échantillon hybridé, mais ils sont ciblés par des sondes qui représentent des hybridations croisées avec un ou plusieurs autres groupes présents dans l'échantillon.

Ces deux possibilités peuvent causer des problèmes lors de l'interprétation biologique des résultats. Nous avons alors décidé d'ajouter une dernière étape à notre programme. L'objectif de cette étape est de vérifier toutes les hybridations croisées possibles qui peuvent être à l'origine d'une mauvaise interprétation des résultats et d'énumérer toutes les interprétations biologiques possibles (les différentes listes possibles de groupes présents dans l'échantillon hybridé).

3.2.2. Optimisation des résultats obtenus par transformation en problème SAT

Dans cette étape, nous utiliserons les résultats des tests de spécificité que nous avons réalisés dans l'étape 1 de notre programme.

D'abord, nous transformons le problème d'interprétation des résultats obtenus aux étapes précédentes de notre programme, en une instance SAT sous forme normale conjonctive CNF. Les littéraux sont les organismes ciblés par la biopuce. Les clauses représentent les hybridations croisées des sondes ciblant les organismes de la liste préliminaire des groupes positifs $\mathcal{L}_G$, déterminée dans l'étape 5 de notre programme. En effet, nous commençons par les groupes qui n'appartiennent pas à $\mathcal{L}_G$ (groupes négatifs ou absents de l'échantillon hybridé). Ces groupes sont transformés en clauses mono-littéral négatives (clauses formées d'un seul littéral qui est négatif). Par exemple un groupe G1 qui n'appartient pas à $\mathcal{L}_G$ est représenté dans l'instance SAT créée par la clause ($\neg$G1). Ensuite, on traite les groupes positifs. Pour chaque groupe $\mathcal{L}_G$ de la liste, nous considérons seulement les régions (ou sondes dégénérées) positives. Puis, pour chaque région positive de ces groupes on choisit la sonde positive qui représente le moins d'hybridations croisées

potentielles, et ce en prenant en considération seulement les groupes de déterminés potentiellement présents dans l'échantillon hybridé (groupes appartenant à la liste $\mathcal{L}_G$). La sonde choisie constitue une clause dont les littéraux sont le groupe ciblé par cette sonde ainsi que les groupes pouvant présenter des hybridations croisées avec elle.

Ainsi, soit une biopuce qui contient des sondes ciblant 5 groupes d'organismes à savoir G1, G2, G3, G4, G5. Soit $\mathcal{L}_G$ la liste préliminaire des groupes potentiellement présents dans l'échantillon hybridé tel que: $\mathcal{L}_G$ = {G1,G2,G3}. Pour construire la CNF correspondant à ce problème, nous commençons par les groupes négatifs qui n'appartiennent pas à $\mathcal{L}_G$. Deux premières clauses sont alors créées: (¬G4) et (¬G5). Nous traitons ensuite les groupes positifs appartenant à $\mathcal{L}_G$.

La région $R_{1,1}$ du groupe G1 est constituée des sondes S1 et S2 qui sont toutes les deux positives. Soient $\mathcal{H}_{S1}$ et $\mathcal{H}_{S2}$ respectivement les ensembles des hybridations croisées des sondes S1 et S2, tels que: $\mathcal{H}_{S1}$ = {G2,G3} et $\mathcal{H}_{S2}$ = {G2,G4,G5}. Ici S1 possède un nombre d'hybridations croisées égal à 2 et est donc inférieur à celui de S2 (3 hybridations). Cependant, si on considère seulement les groupes de $\mathcal{L}_G$, la sonde S1 possède 2 hybridations croisées et S2 possède 1 seule. On choisit donc de représenter cette région par la sonde 2. La clause suivante est alors créée: (G1 ∨ G2 ∨ G4 ∨ G5).

La région $R_{1,2}$ du groupe G1 est constituée des sondes S3 et S4. Seulement S3 est positive et possède une seule hybridation croisée potentielle $\mathcal{H}_{S3}$ = {G2}. La clause suivante est alors créée pour la région $R_{1,2}$: (G1 ∨ G2).

Le groupe G2 contient une seule région $R_{2,1}$. Cette région est constituée de deux sondes positives S5 et S6 qui ne possèdent aucune hybridation croisée potentielle $\mathcal{H}_{S5}$ = Ø et $\mathcal{H}_{S6}$ = Ø. La clause suivante est alors créée pour $R_{2,1}$: (G2).

Le groupe G3 contient une seule région $R_{3,1}$. Cette région est constituée d'une seule sonde positive S7 qui ne possède aucune hybridation croisée potentielle $\mathcal{H}_{S7}$ = Ø. La clause suivante est alors créée pour $R_{3,1}$: (G3).

Nous avons ainsi construit la CNF suivante qui représente notre problème d'interprétation des résultats de cette biopuce:

$\mathcal{F}$ = (¬G4) ∧ (¬G5) ∧ (G1 ∨ G2 ∨ G4 ∨ G5) ∧ (G1 ∨ G2) ∧ (G2) ∧ (G3).

Enfin, nous utilisons le solveur SAT zchaff (Zhang et al., 2001) pour déterminer toutes les solutions à ce problème. Ce solveur complet permet de déterminer si une formule $\mathcal{F}$ sous forme CNF est satisfaisable ou non. Il vérifie s'il existe un modèle $\mathcal{I}$ pour $\mathcal{F}$. Si c'est le cas, zchaff indique le premier modèle trouvé mais ne propose pas tous les modèles possibles. Aussi, pour avoir tous les modèles possibles qui satisfassent $\mathcal{F}$, nous avons développé un programme qui exécute zchaff plusieurs fois. Après chaque modèle trouvé, notre programme récupère le modèle trouvé et insère, dans $\mathcal{F}$, une clause de blocage qui nous évite de retrouver le même modèle une autre fois. Cette clause représente la négation de l'interprétation $\mathcal{I}$ trouvée. Par exemple si $\mathcal{I}$ = {G1=faux, G2=vrai, G3=faux} alors la clause de négation (ou de blocage) que notre programme insère dans $\mathcal{F}$ est la suivante:

$\neg (\neg G1 \wedge G2 \wedge \neg G3) = (G1 \vee \neg G2 \vee G3)$

Notre programme s'arrête lorsque l'insertion de nouvelles clauses rend $\mathcal{F}$ insatisfaisable, c'est-à-dire qu'il n'existe plus aucun modèle qui la satisfasse. Il remplace enfin les noms des variables propositionnelles utilisées dans $\mathcal{F}$ par les noms des groupes correspondants et enregistre le résultat final. Notre programme donne aussi en résultats les différentes valeurs et réponses calculées dans ses premières étapes, pour les groupes, les régions, les sondes et les sondes de contrôle négatives.

Ainsi, si nous reprenons le problème de l'exemple 3, la formule CNF que nous avons déjà construite pour cet exemple est:

$\mathcal{F} = (\neg G4) \wedge (\neg G5) \wedge (G1 \vee G2 \vee G4 \vee G5) \wedge (G1 \vee G2) \wedge (G2) \wedge (G3)$.

Notre programme va ensuite résoudre ce problème. Un premier modèle est trouvé: $\mathcal{I}$ = {G1=vrai, G2=vrai, G3=vrai, G4=faux, G5=faux}. Le premier résultat obtenu est alors le suivant (figure 46):

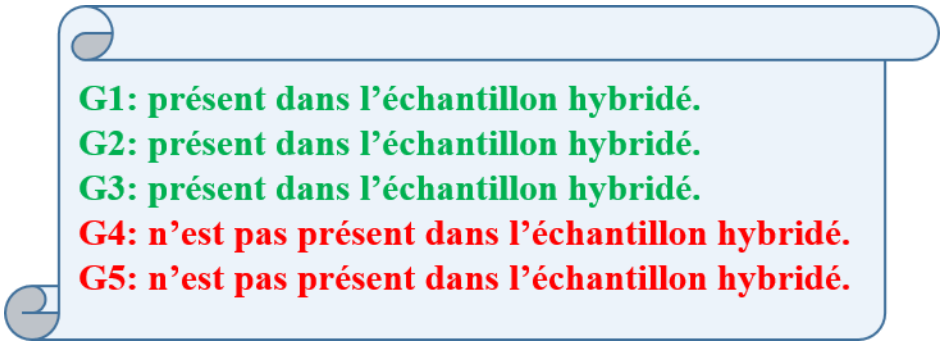

Figure 46: La première solution trouvée pour l'exemple proposé.

Ce modèle est le même que la liste initiale des groupes positifs $\mathcal{L}_G$ que nous avons obtenue avant la transformation de notre problème en une CNF. Nous insérons ensuite une clause de blocage pour le modèle trouvé, dans $\mathcal{F}$. Dans ce cas, cette clause est:

$$\neg(G1 \wedge G2 \wedge G3 \wedge \neg G4 \wedge \neg G5) = (\neg G1 \vee \neg G2 \vee \neg G3 \vee G4 \vee G5)$$

La nouvelle formule $\mathcal{F}'$ ainsi construite est:

$$\mathcal{F}' = (\neg G4) \wedge (\neg G5) \wedge (G1 \vee G2 \vee G4 \vee G5) \wedge (G1 \vee G2) \wedge (G2) \wedge (G3) \wedge (\neg G1 \vee \neg G2 \vee \neg G3 \vee G4 \vee G5).$$

Nous terminons ensuite le traitement. Un nouveau modèle qui satisfait la CNF $\mathcal{F}'$ est alors trouvé: $\mathcal{I}'$ = {G1=faux, G2=vrai, G3=vrai, G4=faux, G5=faux}. Ce problème correspond à une nouvelle solution que nous n'avons pas détectée sans l'utilisation de notre approche de transformation en problème SAT. Cette nouvelle solution est la suivante (figure 47):

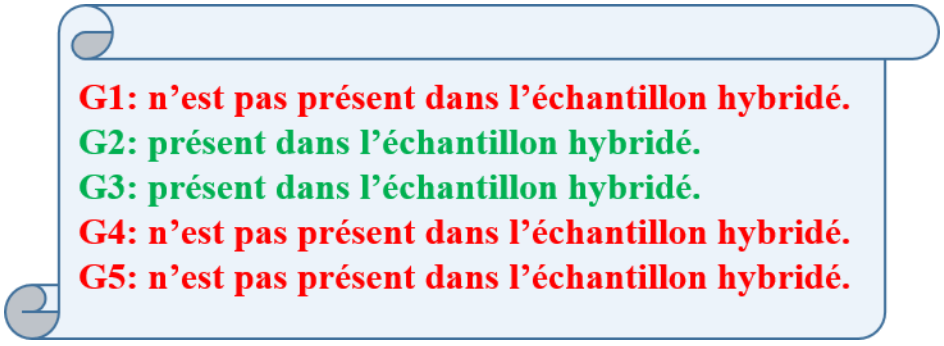

Figure 47: La deuxième solution trouvée pour l'exemple donné.

Nous insérons ensuite une nouvelle clause de blocage pour le deuxième modèle trouvé. Cette clause est:

¬(¬G1 ∧ G2 ∧ G3 ∧ ¬G4 ∧ ¬G5) = (G1 ∨ ¬G2 ∨ ¬G3 ∨ G4 ∨ G5)

La nouvelle formule $\mathcal{F}''$ construite est:

$\mathcal{F}''$ = (¬G4) ∧ (¬G5) ∧ (G1 ∨ G2 ∨ G4 ∨ G5) ∧ (G1 ∨ G2) ∧ (G2) ∧ (G3) ∧ (¬G1 ∨ ¬G2 ∨ ¬G3 ∨ G4 ∨ G5) ∧ (G1 ∨ ¬G2 ∨ ¬G3 ∨ G4 ∨ G5).

Notre programme essaie alors de chercher un nouveau modèle. Néanmoins, zchaff trouve que F'' est insatisfaisable. Il n'existe donc aucune autre solution pour notre problème. Notre programme s'arrête alors et enregistre les deux solutions trouvées.

3.3. Résultats

3.3.1. Résultats de la parallélisation de l'étape du calcul de la liste des hybridations croisées des sondes

La première étape de notre logiciel consiste à tester la spécificité de toutes les sondes utilisées sur la biopuce et de déterminer, à partir des résultats de ce test, la liste des hybridations croisées de chaque sonde. Pour les biopuces de haute densité, cette étape peut nécessiter un temps de calcul considérable. Nous avons alors parallélisé cette tâche pour l'exécuter sur un cluster de calcul. Nous avons testé la performance de cette parallélisation en utilisant un cluster de calcul de 22 nœuds bi-processeurs Quad Core AMD Opteron à 2,1 GHz sous Linux. Le test a été réalisé sur une biopuce de la marque Agilent[3], composée de toutes les sondes sélectionnées à l'aide de KASpOD (Parisot et al., 2012) dans le cadre de la construction de la base de données PhylOPDb (Jaziri et al., 2014b) (voir chapitre 4 section 3 pour plus de détail). Elle contient donc un total de 54 129 sondes 25-mers ciblant 1 295 genres procaryotes. Cette tâche nécessite environ 9 jours pour être exécutée sur un seul cœur d'un processeur AMD Opteron à 2,1 GHz. Les résultats présentés dans la figure 48 montrent que la parallélisation de cette étape permet de réduire considérablement le temps de calcul avec un speedup d'environ 28x en utilisant 32 cœurs de calcul (tâche réalisée en environ 4 heures de calcul) et d'environ 52x en utilisant 64 cœurs de calcul (tâche réalisée en moins de 8 heures de calcul).

[3] http://www.genomics.agilent.com/

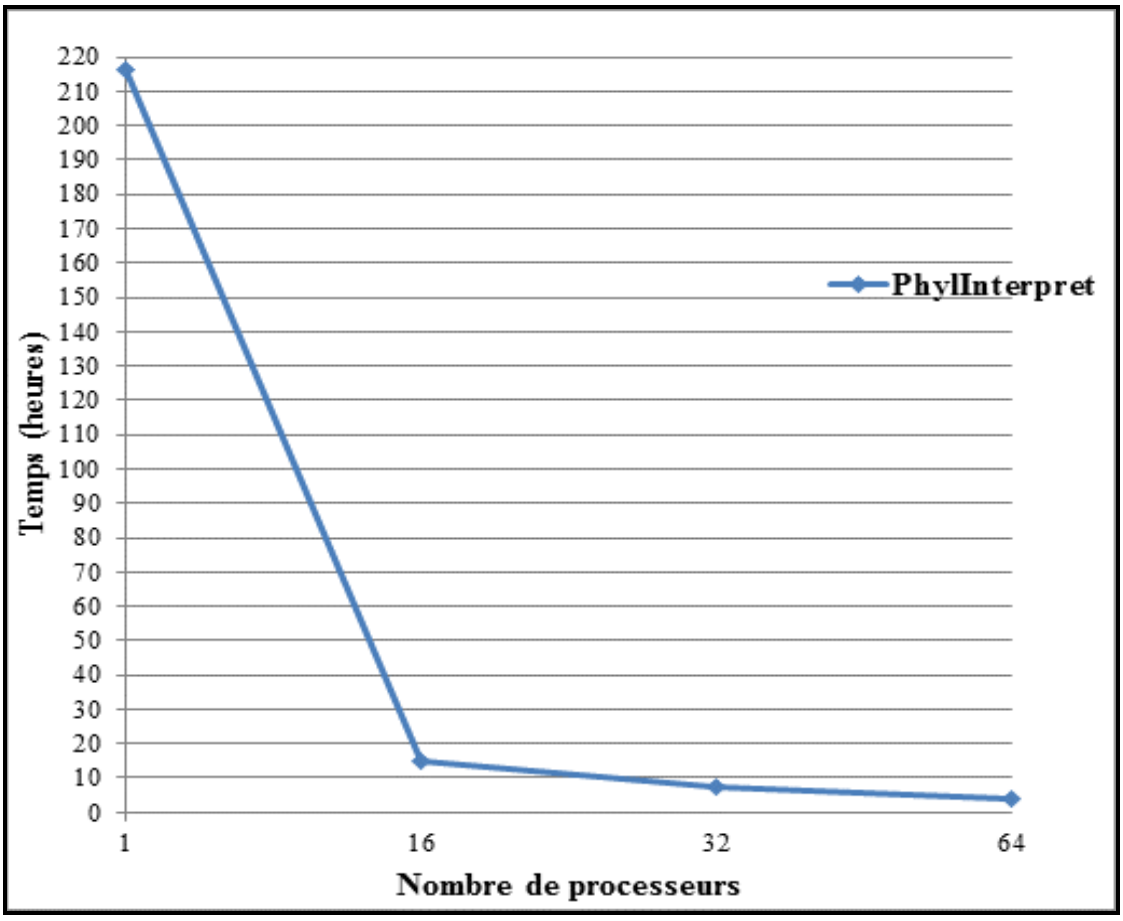

Figure 48: Résultats de l'exécution sur un cluster de calcul de l'étape de détermination des hybridations croisées pour la biopuce procaryote développée avec KASpOD.

3.3.2. Résultats de l'approche d'utilisation des contrôles négatifs pour le calcul d'une valeur minimale de réponse des sondes

Notre programme PhylInterpret utilise la valeur du bruit de fond local correspondant à chaque sonde pour calculer une valeur SNR (« Signal to Noise Ratio ») permettant de déterminer la réponse de la sonde. Cependant, dans certains cas, le logiciel utilisé par l'utilisateur pour l'extraction des données numériques à partir des images de biopuces ne permet pas de calculer le bruit de fond. Pour cela nous avons proposé une méthode qui permet de calculer une valeur minimale de réponse pour chaque sonde, en se basant sur les intensités des contrôles négatifs et de leurs positions sur la biopuce. La valeur calculée remplace celle du bruit de fond local dans le calcul du SNR.

Nous avons testé cette méthode pour calculer le bruit de fond local (ou valeur minimale de réponse) pour toutes les sondes d'une biopuce fonctionnelle de la marque NimbleGen[4] développée avec HiSpOD (Dugat-Bony et al., 2011). Cette biopuce contient 134 704 sondes (16 838 sondes différentes avec 8 réplicats pour chacune) et 5 707 contrôles négatifs. Le temps de calcul nécessaire pour effectuer cette tâche est de 441 secondes (7 min 21s) en utilisant un seul cœur sur un processeur AMD Opteron à 2,3 GHz sous Linux. Ce temps reste raisonnable mais peut être beaucoup plus élevé pour des biopuces contenant jusqu'à des millions de sondes ou dans le cas de l'analyse de plusieurs biopuces simultanément.

3.3.3. Résultats de l'approche de détermination de la structure procaryotique de l'échantillon biologique hybridé

Notre algorithme d'analyse des données issues des biopuces à ADN consiste à déterminer la composition (structure) en microorganismes procaryotes d'un échantillon biologique hybridé. Tout d'abord, les intensités de toutes les sondes sont calculées et le seuil de réponse déterminé par l'utilisateur permet de déterminer une liste initiale de tous les groupes microbiens pouvant être présents dans l'échantillon biologique hybridé. Puis, le programme transforme le problème d'analyse des résultats de la biopuce en un problème SAT (problème de satisfaisabilité) en utilisant le langage propositionnel. Il utilise ensuite le solveur SAT que nous avons retenu, à savoir zchaff (Zhang et al., 2001), et notre méthode d'insertion de clauses bloquantes pour calculer toutes les interprétations possibles des résultats de l'hybridation de la biopuce. Nous avons testé notre approche pour l'analyse des données issues de l'hybridation d'une biopuce de la marque Agilent[5] composée de l'ensemble de sondes 25-mers sélectionnées avec PhylGrid2.0 (Jaziri et al., 2014a) dans le cadre de la construction de la base PhylOPDb (voir chapitre 4 section 3). Elle contient 19 874 sondes ciblant 2 069 genres procaryotes. Pour chaque sonde, 5 réplicats sont présents sur la biopuce ADN. La biopuce contient alors un total de 99 370 sondes. Les valeurs du bruit de fond sont données en entrée après avoir été calculées comme précédemment indiqué tout comme l'étape de détermination des hybridations croisées (tâche réalisée dans la section résultats 3.4.1). Nous avons utilisé notre programme en faisant varier différents paramètres d'entrée. Le tableau 13 illustre les résultats en nombre

[4] www.nimblegen.com/

[5] http://www.genomics.agilent.com/

de solutions trouvées et en temps de calcul en utilisant un cœur de calcul d'un processeur AMD Opteron à 2,3 GHz sous Linux (sans ajouter le temps nécessaire pour le calcul des hybridations croisées des sondes). Ces résultats montrent que notre logiciel permet de produire l'ensemble des solutions en un temps raisonnable qui ne dépasse pas les 4 minutes pour l'analyse de cette biopuce quels que soient les paramètres utilisés.

L'échantillon hybridé sur cette biopuce est composé d'un mélange d'ADN d'espèces connues appartenant à 31 genres procaryotes. Bien que nous ayons constaté que l'hybridation n'était pas parfaite du fait d'un problème de fabrication de la biopuce (la biopuce a été re-fabriquée par le fournisseur, les nouvelles hybridations sont en cours), les résultats d'analyse obtenus avec PhylInterpret sont prometteurs. En effet, pour vérifier la qualité des résultats produits par notre logiciel, nous avons retenu l'analyse numéro 3 du tableau 13 (3ème ligne du tableau 13). Pour cette analyse, nous avons utilisé un seuil SNR égal à 5 pour déterminer la réponse des sondes et un pourcentage minimal de régions positives d'un genre égal à 60% (3 régions sur 5).

Tableau 13: Résultats de l'analyse de la biopuce procaryote composée de 19 874 sondes 25-mers sélectionnées avec PhylGrid2.0: « Seuil réponse sondes » correspond au seuil SNR minimal pour considérer qu'une sonde a une réponse positive. « Seuil réponse groupes » correspond au pourcentage minimal de régions positives du groupe pour le considérer comme présent dans l'échantillon hybridé.

Seuil réponse sondes	Seuil réponse groupes (%)	Nb solutions trouvées	Temps de calcul
10	60	32	3 m 17.991 s
10	80	4	3 m 22.029 s
5	60	128	3 m 33.256 s
5	80	4	3 m 57.623 s

Dans cet exemple, la plus grande liste de genres produite avec PhylInterpret contient un total de 63 genres. La plus petite liste contient 57 genres. Les 6 genres de différence entre les deux listes sont des genres qui ne sont pas présents dans l'échantillon hybridé mais qui apparaissent du fait d'hybridations croisées. PhylInterpret a permis donc de supprimer ces genres en prenant en compte la spécificité des sondes de la biopuce. Cela

démontre dès à présent la bonne qualité des résultats obtenus en utilisant notre programme et aussi la qualité des sondes déterminées avec notre logiciel de sélection de sondes PhylGid2.0, même en présence des problèmes d'hybridation constatés sur la biopuce. En effet, très peu d'hybridations faussement positives sont détectées (moins de 2% des genres ciblés par la biopuce). Nous ne développerons pas plus les interprétations des résultats sachant qu'il existe un problème de fabrication des biopuces.

4. Discussion

Nous avons développé PhylInterpret dans le but d'analyser les données issues de biopuces à ADN développées avec MetaExploArrays (chapitre 5), PhylGrid2.0 (chapitre 4), KASpOD (Parisot et al., 2012) ou PhylArray (Militon et al., 2007). Il est actuellement le seul logiciel disponible adapté à traiter les résultats de biopuces aussi complexes. Les rares logiciels permettant de déterminer les groupes taxonomiques présents dans un échantillon donné tels que PhyloTrac (Schatz et al., 2010) ne sont pas compatibles pour résoudre de telles complexités. En effet, PhyloTrac est spécifique à l'analyse des biopuces PhyloChip qui utilisent des couples de sondes PM (« perfect match probe») et MM (« mismatch probe») et ne peut donc pas être utilisé sur d'autres types de biopuces. De plus, PhylInterpret est, à notre connaissance, le seul logiciel d'analyse de biopuces qui offre la possibilité de vérifier la spécificité des sondes de la biopuce et d'exploiter ces informations de spécificité pour permettre une meilleure interprétation biologique des résultats.

Afin de déterminer la composition d'un échantillon suite à l'hybridation sur une biopuce à ADN, PhylInterpret cherche toutes les solutions possibles, en fonction de la spécificité des sondes utilisées. PhylInterpret transforme alors le problème d'analyse des résultats de biopuces en une instance du problème de Satisfaisabilité, en se basant sur le langage de la logique propositionnelle. Il utilise ensuite un solveur SAT puissant, à savoir zchaff (Zhang et al., 2001), avec un script d'insertion des clauses de blocage pour chaque modèle trouvé, pour déterminer la liste complète et non redondante des solutions. Cependant, le nombre de solutions (ou listes) générées par notre programme dépend fortement de la qualité de la biopuce et du niveau de spécificité des sondes utilisées. Dans le cas d'un échantillon biologique complexe, si de nombreuses sondes représentent un grand nombre d'hybridations croisées, le nombre de solutions produites peut être très élevé. Cela peut poser des difficultés d'interprétation et de prise de décision pour les

biologistes. Toutefois, quelles que soient la nature et la qualité des biopuces, la génération et la vérification de toutes les solutions possibles restent obligatoires pour faciliter les interprétations. D'où l'intérêt de l'utilisation de notre programme même dans le cas où il génère un grand nombre de solutions possibles.

Notre approche requiert donc la réalisation d'un test de spécificité de toutes les sondes utilisées sur la biopuce. Cependant, pour les résultats issus des biopuces de haute densité, cette étape devient très coûteuse en temps de calcul (plusieurs jours de calcul). Pour cela nous avons parallélisé ce traitement en le partageant équitablement sur plusieurs jobs qui seront exécutés parallèlement sur un cluster de calcul.

Pour déterminer la nature et le niveau de réponse des sondes, nous utilisons une approche d'estimation de la valeur minimale de réponse d'une sonde. Cette valeur peut également être utilisée pour mesurer le niveau d'intensité des hybridations non spécifiques sur la biopuce, et pour contrôler la qualité de l'expérience d'hybridation réalisée. Notre approche se base sur les intensités d'hybridation des sondes de contrôle négatives et leurs positions par rapport aux autres sondes composant la biopuce. Ainsi, cette approche dépend fortement de la position et de la qualité des contrôles négatifs utilisés. En effet, ces contrôles doivent être dispersés de façon régulière sur toute la surface de la biopuce et ne doivent pas s'hybrider parfaitement avec des séquences de l'échantillon hybridé. Pour cela, dans nos travaux futurs, nous envisageons le développement de deux nouveaux algorithmes. Le premier permet de développer des sondes de contrôle négatives pour les biopuces, en fonction des sondes utilisées et de l'échantillon à hybrider. Le deuxième algorithme sert à distribuer toutes les sondes de contrôle utilisées, d'une manière régulière et équitable, sur la surface entière de la biopuce. Nous envisageons également la parallélisation de l'approche de calcul de la valeur minimale de réponse des sondes. Dans ce contexte, l'utilisation des cartes accélératrices pour calculs parallèles comme les processeurs graphiques GPGPUs (« General-purpose computing on graphics processing units ») ou les coprocesseurs Xeon Phi d'Intel[6] peut réduire considérablement la complexité et le temps de calcul de cette approche surtout dans le cas de l'analyse simultanée de plusieurs biopuces de haute densité.

[6] http://www.intel.fr/content/www/fr/fr/processors/xeon/xeon-phi-detail.html

5. Conclusion

Dans ce chapitre, nous avons présenté PhylInterpret un nouveau logiciel d'analyse des données issues des expériences d'hybridation des biopuces à ADN. PhylInterpret utilise des notions de la logique propositionnelle (problème de Satisfaisabilité) pour déterminer la composition de l'échantillon hybridé en énumérant tous les ensembles possibles de groupes présents dans cet échantillon. Ainsi, les résultats obtenus montrent qu'avec l'échantillon biologique hybridé sur la biopuce, qui n'était cependant pas d'une qualité optimale, nous avons pu identifier la plupart des microorganismes composant cet échantillon.

Dans le chapitre suivant, nous présentons un programme distribué pour la traduction inverse complète d'oligopeptides, appliqué à la sélection de sondes pour biopuces à ADN fonctionnelles.

Chapitre 7: Approche parallèle pour la traduction inverse complète d'oligopeptides pour la conception des biopuces fonctionnelles

1. Introduction

La traduction inverse complète des séquences protéiques a récemment été utilisée pour initier la sélection de sondes pour biopuces à ADN fonctionnelles à partir des régions peptidiques conservées. Cette stratégie peut contribuer à mieux évaluer la totalité de la diversité génétique microbienne présente dans les environnements complexes. Dans ce chapitre, nous présentons un nouvel algorithme distribué pour le calcul de la traduction inverse complète des oligopeptides courts et pour le filtrage des séquences nucléotidiques générées avec les critères habituels de sélection de sondes pour biopuces fonctionnelles. Nous utilisons une approche de méta-programmation et d'ingénierie dirigée par les modèles pour traiter simultanément plusieurs centaines d'oligopeptides. Cette approche d'ingénierie des modèles utilise d'une part la méta-programmation, et d'autre part la production (par transformation) de code adapté pour différents moyens de calcul: PC, multiprocesseurs de type SMP (« Symmetric Multi-Processor »), clusters ou grille de calcul. Nous présentons les performances et les statistiques de notre algorithme sur des jeux de données biologiques réels et simulés.

2. Généralités et motivations

L'étape biologique qui consiste à produire une séquence protéique correspondante à une séquence d'ARNm est appelée traduction. En revanche, la traduction inverse complète est l'étape permettant de définir l'ensemble des séquences d'acides nucléiques possibles à partir d'une séquence protéique. Il s'agit d'une étape non biologique, mais purement algorithmique. C'est grâce à la découverte du mécanisme de synthèse des protéines, suivie par le décodage complet du code génétique en 1961 par le prix Nobel M. Nirenberg, qu'il a été établi qu'une même séquence protéique peut être codée par plusieurs séquences nucléiques (Nirenberg, 2004). En effet, chaque acide aminé peut être codé par deux, trois, quatre ou six codons[1], à l'exception de la *Méthionine* (notée M) et du *Tryptophane* (notée W), qui ne sont codé que par un seul codon. Étant donné que le codon est un triplé de nucléotides et que 4 nucléotides différents existent (l'*Adénine* « A », la *Thymine* « T », la *Cytosine* « C » et la *Guanine* « G »), il est donc possible de définir 4^3 soit 64 codons différents (tableau 14). La traduction inverse complète des séquences d'acides aminés représente alors une tâche fastidieuse qui peut générer des quantités massives de données.

[1] Codon: groupe de trois acides nucléiques.

Par exemple, un seul oligopeptide de 15 acides aminés peut fournir jusqu'à environ 5.10^{11} séquences ADN possibles d'une longueur de 45 nucléotides, et peut nécessiter jusqu'à plus de 19 To d'espace disque pour le stockage des séquences obtenues. Cependant, dans le contexte de découverte de nouveaux variants géniques, il est important de pouvoir identifier toutes les séquences nucléotidiques pouvant coder une protéine donnée (Dugat-Bony et al., 2012b). Dans ce contexte, l'utilisation de la traduction inverse pour le développement de sondes pour biopuces à ADN est une approche très prometteuse.

Tableau 14. Liste des acides aminés avec leurs codes et leurs codons correspondants.

Acide aminé	Code	Codons
Isoleucine	I	ATT, ATC, ATA
Leucine	L	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	GTT, GTC, GTA, GTG
Phenylalanine	F	TTT, TTC
Methionine	M	ATG
Cysteine	C	TGT, TGC
Alanine	A	GCT, GCC, GCA, GCG
Glycine	G	GGT, GGC, GGA, GGG
Proline	P	CCT, CCC, CCA, CCG
Threonine	T	ACT, ACC, ACA, ACG
Serine	S	TCT, TCC, TCA, TCG, AGT, AGC
Tyrosine	Y	TAT, TAC
Tryptophan	W	TGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAT, AAC
Histidine	H	CAT, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAT, GAC
Lysine	K	AAA, AAG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Stop codons	* (Stop)	TAA, TAG, TGA

De nouvelles stratégies ont été proposées pour initier la sélection de sondes pour biopuces fonctionnelles FGAs à partir des séquences peptidiques conservées. Ces stratégies utilisent la traduction inverse complète des oligopeptides. En effet, il est important d'avoir tous les variants de séquences nucléotidiques correspondant à une protéine d'intérêt lors de l'étude des environnements complexes qui sont composés principalement de microorganismes inconnus. Dans le cas des biopuces à ADN, l'objectif est de traiter uniquement des courtes séquences de protéines appelées « oligopeptides » mais représentant une signature spécifique de la protéine ciblée. En pratique, la taille des sondes utilisées dans les biopuces à ADN est souvent comprise entre 20 et 50 nucléotides. Ces sondes peuvent être obtenues par la traduction inverse des oligopeptides courts de 7 à 17 acides aminés. Dans ce cas, la traduction inverse complète devient une tâche bornée et raisonnable, surtout avec l'utilisation d'architectures parallèles et distribuées qui peuvent réduire considérablement le temps de calcul. En outre, la sélection de sondes oligonucléotidiques pour biopuces fonctionnelles doit répondre à certains critères principaux tels que la spécificité, la sensibilité et l'uniformité. L'utilisation de ces critères pour filtrer les sondes générées par une traduction inverse complète peut, d'une part, améliorer la qualité des résultats, et d'autre part, réduire considérablement les quantités de données générées et transmises avec un système distribué. Ainsi, l'espace disque nécessaire pour sauvegarder uniquement les séquences finales filtrées devient beaucoup plus raisonnable, surtout avec les capacités de stockage actuelles.

Ici, nous présentons une nouvelle méthode de parallélisation pour le calcul de la traduction inverse complète d'oligopeptides pour le développement des biopuces à ADN fonctionnelles. Notre méthode permet la génération de tous les oligonucléotides possibles à partir d'un ensemble d'oligopeptides courts. Les oligonucléotides obtenus sont ensuite filtrés avec des critères habituels de sélection de sondes pour biopuces. L'algorithme que nous proposons traite spécifiquement les oligopeptides composés de 7 à 17 acides aminés, même s'il peut traiter des peptides plus longs avec une dégénérescence inférieure. Nous utilisons les approches d'ingénierie dirigée par les modèles et de méta-programmation pour générer automatiquement les codes sources nécessaires à la gestion et le déploiement de l'algorithme sur différentes platesformes: PCs, multiprocesseurs (SMP), clusters de calcul ou grilles de calculs comme la grille de calcul européenne EGI que nous utilisons dans ce travail pour traiter les tâches les plus lourdes.

3. Aperçu des travaux existants

La traduction inverse d'une séquence protéique se réfère souvent à fournir une séquence d'ADN qui code pour cette protéine chez l'organisme étudié. Les applications qui utilisent la traduction inverse ont introduit différentes solutions pour obtenir les séquences nucléotidiques les plus probables parmi toutes les possibilités. Par exemple, ces applications peuvent utiliser une méthode basée sur la sélection du codon le plus fréquent pour chaque acide aminé (biais d'usage des codons chez un organisme donné), ou une méthode favorisant la réutilisation des codons (Missaoui et al., 2007). Bien que de telles méthodes soient rapides, l'utilisation préférentielle de certains codons par rapport à d'autres introduit une perte d'informations considérable, surtout si l'objectif est d'étudier des variants de séquences sans aucun a priori.

La traduction inverse est utilisée dans plusieurs applications nécessitant un passage de protéines à l'ADN ou à l'ARN, telles que l'alignement des séquences au niveau protéique. Dans ce contexte, un programme appelé « RevTrans » a été proposé pour construire un alignement multiple d'ADN à partir d'un alignement de séquences protéiques (Wernersson and Pedersen, 2003). Tout d'abord, RevTrans traduit un ensemble de séquences nucléotidiques en séquences d'acides aminés. Ensuite, il aligne les séquences protéiques obtenues et effectue enfin une traduction inverse des séquences alignées pour construire un alignement multiple de séquences ADN dans lequel les positions de codon analogues sont donc toujours alignées. Dans cette méthode, chaque acide aminé est tout simplement remplacé par son codon dans la séquence nucléotidique de départ.

Gîrdea et al. (2010), ont également utilisé la traduction inverse pour trouver des homologies entre protéines distantes ayant subi des mutations entraînant des décalages de cadre de lecture (frameshift). La méthode proposée effectue un alignement de séquences nucléiques contenant des frameshifts, par traduction inverse des protéines en utilisant une transformation en graphes de toutes les séquences ADN codantes de chaque protéine. Un algorithme de programmation dynamique utilise un système de scores pour déterminer les deux séquences ADN putatives qui ont le meilleur score d'alignement. Dans cette application, la traduction inverse consiste à représenter toutes les séquences d'ADN codantes possibles pour une protéine par un graphe acyclique orienté. Dans ce graphe, chaque branche représente une séquence possible et chaque nœud, à une position « *i* », représente un nucléotide possible qui peut apparaître à la position « *i* » dans au moins une

des séquences codantes. Un ensemble de nœuds existe à chaque position « *i* » du graphe. Le nombre de nœuds est réduit en utilisant le code IUPAC de nucléotides dégénérés. Avec cette approche, toutes les séquences codantes sont explorées, mais la traduction inverse ne donne pas une énumération explicite de toutes les séquences nucléotidiques possibles.

En outre, certaines méthodes ont été proposées pour effectuer la traduction inverse par la recherche des séquences nucléotidiques dérivées directement à partir d'une séquence d'acides aminés, en se basant sur les codes IUB/IUPAC des nucléotides dégénérés (Stothard, 2000). Cette approche consiste à représenter toutes les séquences possibles générées à partir d'une protéine donnée, par une seule séquence consensus. En effet, chaque acide aminé est traduit à un codon IUB/IUPAC unique qui représente tous les codons possibles par un seul triplet de nucléotides. Le tableau 15 contient les triplets de nucléotides dégénérés qui représentent les codons de tous les acides aminés en utilisant les codes IUB/IUPAC des nucléotides dégénérés. Les programmes qui utilisent cette technique requièrent seulement le remplacement de chaque acide aminé par son codon IUB/IUPAC correspondant.

La traduction inverse d'une protéine peut également être effectuée par le calcul de la séquence ADN en considérant la préférence d'usage des codons pour chaque organisme (Nash, 1993; Pesole et al., 1988). La plupart des outils de traduction inverse proposés s'appuient sur le calcul des probabilités des codons d'acides aminés en se référant aux banques de données nucléotidiques. Ces outils sont basés sur l'usage des codons: les fréquences de codons connus de l'espèce cible sont utilisées comme des probabilités d'assignation des codons pour chaque acide aminé. Dans ce contexte, certaines approches proposées utilisent spécialement des algorithmes génétiques (Moreira, 2004; Moreira and Maass, 2004), les réseaux de neurones (White and Seffens, 1998) ou la classification hiérarchique (Ma et al., 2002) pour déterminer le codon le plus probable à une position donnée puis sélectionner la traduction inverse la plus appropriée pour une séquence protéique.

Tableau 15. Les triplés de nucléotides dégénérés représentant tous les codons des acides aminés en utilisant le code IUB/IUPAC.

Acide aminé	Codon consensus
A	**GCN**
C	**TGY**
D	**GAY**
E	**GAR**
F	**TTY**
G	**CGN**
H	**CAY**
I	**ATH**
K	**AAR**
L	**MTN**
M	**ATG**
N	**AAY**
P	**CCN**
Q	**CAR**
R	**MGN**
S	**WSN**
T	**ACN**
V	**CTN**
W	**TGG**
Y	**TAY**
*****	**TAR**

Certains outils qui utilisent les codons les plus probables ou les codons sélectionnés par l'utilisateur pour effectuer la traduction inverse sont aussi disponibles via des interfaces web (« Backtranseq »[2], « Gene Design - Reverse translation »[3], « Sequence Manipulation

[2] http://emboss.bioinformatics.nl/cgi-bin/emboss/backtranseq (consulté décembre 2013).

[3] http://54.235.254.95/cgi-bin/gd/gdRevTrans.cgi (consulté décembre 2013).

Suite - Reverse Translate »[4]). L'utilisation de ces outils permet de simplifier le passage d'une séquence protéique d'intérêt à la séquence nucléotidique correspondante et de diminuer la quantité de séquences nucléotidiques générées par une traduction inverse. En revanche, l'inconvénient majeur est la perte d'information engendrée par une telle pratique. En effet, des séquences nucléotidiques potentiellement codantes pour la protéine traitée peuvent être ignorées par ces méthodes. Le code génétique permet de générer autant de séquences que la dégénérescence totale de la séquence protéique étudiée, contrairement au simple remplacement des acides aminés par certains codons en se basant sur des critères de choix.

C'est en raison de la complexité et de la nature exponentielle du problème de la traduction inverse complète des séquences protéiques, que la plupart des approches proposées ne permet pas l'énumération de toutes les séquences ADN possibles qui codent pour une même protéine. En effet, certaines méthodes essaient de générer la séquence nucléotidique la plus probable en fonction de différentes techniques telles que les préférences d'usage des codons. Alors que d'autres transforment le problème de traduction inverse en la génération d'un graphe ou d'une séquence consensus qui représente toutes les séquences codantes en se basant sur les codes IUB/IUPAC. À notre connaissance, seuls trois logiciels existants permettent de générer toutes les séquences ADN possibles à partir d'un oligopeptide: DegenRev (Missaoui et al., 2007), StackPrt (Hill, 2006) et PRT (Missaoui et al., 2006).

DegenRev (Missaoui et al., 2007) est un programme en langage C, développé dans le but de réaliser la traduction inverse d'oligopeptides courtes pour la conception de sondes oligonucléotides spécifiques à des gènes codant des protéines. Il est basé sur une technique énumérative régulière qui calcule, à chaque étape de la traduction inverse d'un oligopeptide, la dégénérescence de chaque acide aminé pour déterminer les oligonucléotides restants. Au début, chaque oligopeptide est stocké dans un tableau de longueur « *pl* ». Soit *pl* la taille de l'oligopeptide traité. À chaque acide aminé *i* de l'oligopeptide, $i \in [0, pl - 1]$, le nombre total de codons est noté *NC(i)* et la dégénérescence totale est notée *D*.

$$\text{La dégénérescence } D = \prod_{i=0}^{pl-1} NC(i)$$

[4] http://www.bioinformatics.org/sms2/rev_trans.html (consulté décembre 2013).

Pour chaque acide aminé i de l'oligopeptide traité, trois valeurs de dégénérescence sont calculées:

$$\text{La dégénérescence cumulée: } d(i) = \prod_{k=0}^{i} NC(k)$$

$$\text{La dégénérescence restante: } r(i) = \prod_{k=i+1}^{pl-1} NC(k)$$

$$\text{La dégénérescence passée: } c(i) = \prod_{k=1}^{i} NC(k-1);\ c(i) = 1\ si\ i = 0$$

La suite de l'algorithme dépend de l'initialisation de toutes ces variables. En effet, à la fin de l'initialisation, l'étape principale de la traduction inverse commence. DegenRev teste pour chaque acide aminé si la valeur de la dégénérescence cumulée « r(i) » est atteinte. Si c'est le cas, il traite l'acide aminé suivant. Sinon, il répète « c(i) » fois l'étape de traduction inverse du même codon. En utilisant cette approche, l'algorithme réalise un total de *D* boucles sans perdre de temps dans la réinitialisation des variables de dégénérescence. Ici à chaque acide aminé i ∈ [0, *pl* − 1], la dégénérescence est calculée par:

$$D = d(i) * r(i)$$

Finalement, les oligonucléotides obtenus après la traduction inverse de l'oligopeptide sont triés par leurs valeurs respectives de dégénérescence. Le fichier résultat contient l'ensemble des oligonucléotides générés non redondants, dont chacun représente une séquence codante potentielle pour la séquence protéique étudiée.

La complexité de cet algorithme est calculée par:

$$Complexité = D * pl$$

Où D est la dégénérescence de l'oligopeptide et pl est sa longueur.

DegenRev est approprié pour des peptides qui produisent moins de 10^9 séquences nucléiques possibles, indépendamment de la longueur des oligopeptides. Cependant, ce logiciel utilise une grande quantité de RAM et possède une complexité exponentielle. Pour pallier ces inconvénients de DegenRev, une autre approche avait été proposée dans (Hill, 2006) à la suite de la thèse de Sébastien Rimour pour la traduction inverse d'oligopeptides

possédant une dégénérescence plus importante et avec de meilleures performances. Le programme qui implémente cette approche est StackPrt.

StackPrt a été développé en langage C. Il est basé sur une technique de gestion de pile afin d'utiliser une très petite quantité de mémoire vive. StackPrt utilise une approche de méta-programmation et propose donc un générateur de programmes pour écrire automatiquement le code source dédié à la traduction inverse, en fonction de la longueur des oligopeptides à traiter. Une comparaison des algorithmes DegenRev et StackPrt a été présentée dans (Missaoui et al., 2008). Les tests réalisés dans cette étude ont montré que le programme StackPrt est plus efficace, il permet de traiter des tailles d'oligopeptides plus importante et assure de meilleures performances en temps de calcul, croissantes en fonction de la taille des oligopeptides à traiter (tableau 16: tests réalisés sur une machine Linux multiprocesseurs 8 × AMD Opteron Core duo 1,8 GHz).

Tableau 16. Comparaison des performances en temps de calcul de StackPrt et DegenRev (Missaoui et al., 2008).

Taille oligopeptide	Nombre oligonucléotides produits	Temps de calcul (s)		Facteur Speedup (DegenRev/StackPrt)
		DegenRev	StackPrt	
13	764 411 904	566 661	163 773	3,46
14	4 586 471 424	26 042 736	996 896	26,12
15	18 345 885 696	N.C	4 119 544	N.C

Certaines stratégies, comme « CODEHMOP » et « Metabolic Design », ont été proposées pour initier la sélection de sondes pour les biopuces fonctionnelles FGAs à partir de régions peptidiques conservées en utilisant la traduction inverse. La stratégie CODEHMOP (« COnsensus DEgenerate Hybrid Motif Oligonucleotide Probe ») (Bontemps et al., 2005) est basée sur celle du design d'amorces PCR appelée CODEHOP (« COnsensus DEgenerate Hybrid Oligonucleotide Primer ») (Rose et al., 2003). CODEHMOP identifie les motifs d'acides aminés conservés à partir des alignements multiples de séquences protéiques. Ensuite, toutes les séquences nucléotidiques possibles sont produites à partir de la région la plus fortement conservée de chaque motif d'acides aminés, en utilisant une stratégie qui fournit un nombre limité de nucléotides codant pour la séquence d'acides aminés choisie. Les séquences nucléotidiques produites représentent

les sondes dites « hybrides ». Ces sondes sont flanquées par des extrémités 5’ et 3’ fixées pour augmenter leur longueur. L’application de cette stratégie est cependant limitée par le fait qu’elle n’est pas implémentée dans un programme entièrement automatisé. De plus, aucun test de critères de sélection de sonde n’est intégré. En outre, la stratégie de traduction inverse utilisée ne permet pas l’énumération de toutes les séquences nucléotidiques possibles à partir de la séquence d’acides aminés choisie.

En outre, Terrat et al. ont proposé en 2010 un logiciel de sélection de sondes exploratoires pour biopuces fonctionnelles appelé « Metabolic Design ». Ce logiciel utilise le programme StackPrt (Hill, 2006) pour la traduction inverse d’oligopeptides. D’abord, Metabolic Design assure l’extraction des séquences protéines ciblées qui sont ensuite alignées. Pour chaque site moléculaire de l’alignement multiple des protéines, les acides aminés sont rétrotraduits en utilisant StackPrt. Enfin, la spécificité de toutes les séquences nucléotidiques produites par StackPrt est évaluée pour la détermination des hybridations croisées potentielles avec l’ensemble total de séquences ADN pouvant être présentes dans le milieu étudié. Cet outil de sélection de sondes utilise la traduction inverse complète afin d’énumérer toutes les sondes oligonucléotides possibles. Cependant, aucun critère de sélection de sondes n’est vérifié à l’étape de traduction inverse pour filtrer les séquences nucléotidiques générées. Le filtrage des oligonucléotides par les critères habituels de détermination de sondes pour biopuces, tels que la température de fusion tm, la présence d’homopolymères et le pourcentage des nucléotides G+C, peut considérablement réduire l’énorme quantité de données générées par la traduction inverse complète et améliorer la qualité des résultats. De plus, en pratique, lorsqu’il s’agit de la traduction inverse d’oligopeptides pour le développement des biopuces à ADN fonctionnelles, nous devons traiter simultanément jusqu’à plusieurs milliers d’oligopeptides. Or, l’étape de traduction inverse complète peut être excessivement gourmande en ressources (espace disque et temps de calcul) lorsqu’elle est appliquée à un grand nombre d’oligopeptides. Le calcul parallèle constitue alors une piste bien adaptée pour réduire la complexité de cette tâche et diminuer son temps de calcul. Dans ce contexte de sélection de sondes, un programme parallèle pour la traduction inverse complète d’oligopeptides a été proposé dans (Missaoui, 2006). Ce programme, appelé PRT, permet une traduction inverse à haut débit de protéines non redondantes en utilisant un cluster de calcul afin de réduire le temps d’exécution. Les résultats sont écrits et distribués sur les disques de tous les nœuds de calcul utilisés pour améliorer la performance du programme et éviter les conflits d’accès au disque s’ils étaient

directement centralisés. PRT a été développé en utilisant le langage C et l'implémentation MPICH du standard MPI (« Message Passing Interface ») pour le développement d'applications parallèles[5].

Dans le logiciel PRT, MPI a été seulement utilisé pour partager la charge de calcul sur plusieurs processus. Deux stratégies différentes pour réaliser la traduction inverse complète ont été testées et comparées dans le cadre du développement de PRT: l'allocation dynamique de la mémoire et le mappage des fichiers en mémoire. Cependant, quelle que soit la stratégie utilisée, PRT consomme une grande quantité de mémoire vive et son utilisation est donc limitée par la taille de la mémoire vive du système. De plus, PRT possède une complexité exponentielle malgré l'utilisation de la parallélisation qui a permis de diviser la complexité du problème. L'utilisation de MPI a également induit une limitation de la performance du programme PRT. En effet, l'utilisation de MPI avec un nombre élevé de processus provoque une augmentation significative du temps de communication qui peut être important par rapport au temps de calcul. Pour les grandes tâches (traduction inverse d'un grand nombre d'oligopeptides), cela peut conduire à une importante diminution de la performance. Ici, comme dans de nombreux programmes parallèles communicants, l'accélération du programme PRT est donnée par $log2(N)$, où N est le nombre de processeurs utilisés. En outre, le programme PRT énumère toutes les sondes oligonucléotidiques possibles correspondantes à un oligopeptide donné pour le développement des biopuces fonctionnelles. Cependant, il ne prend pas en compte la qualité de ces sondes qui peuvent être filtrées par les critères habituels de sélection dans ce contexte de détermination de sondes pour FGAs. La fonction principale de PRT est également basée sur l'algorithme DegenRev. Cependant, les tests de performance menés par Missaoui et al. (2008) ont montré que l'utilisation de l'algorithme « StackPrt » est préférable et que depuis l'introduction de ce nouvel algorithme (Hill, 2006), il n'y a plus de raison d'utiliser DegenRev. Pour cela, dans ce chapitre, nous proposons une nouvelle approche parallèle basée sur l'algorithme séquentiel StackPrt que nous avons adapté à la tâche de sélection de sondes pour biopuces fonctionnelles FGAs.

[5] Aa http://www.mcs.anl.gov/research/projects/mpi/

4. P-MetaStackPrt: algorithme distribué pour le calcul de la traduction inverse complète des oligopeptides pour biopuces FGAs

L'algorithme que nous décrivons ci-après doit générer chaque oligonucléotide possible à partir des séquences d'acides aminés, et de filtrer les oligonucléotides obtenus avec des critères de sélection de sondes afin d'adapter les résultats aux exigences du design de sondes pour les biopuces à ADN fonctionnelles. Cet algorithme doit également profiter des avantages du calcul parallèle et distribué pour effectuer la traduction inverse complète d'un grand nombre d'oligopeptides simultanément.

Dans cette section, nous rappelons les principes de l'algorithme séquentiel StackPrt sur lequel se base la fonction principale de notre programme. Puis, nous citons les améliorations que nous avons apportées à cet algorithme et enfin nous exposons les techniques de distribution utilisées.

4.1. L'algorithme original StackPrt

StackPrt (Hill, 2006) est un algorithme séquentiel développé pour effectuer la traduction inverse complète de courts oligopeptides. Il est basé sur une pile qui limite le nombre d'écritures dans la mémoire afin d'absorber autant que possible la nature exponentielle du problème. La profondeur de la pile et le nombre de boucles imbriquées utilisées dans cet algorithme dépendent de la longueur des oligopeptides à traiter. La quantité de RAM utilisée par StackPrt est très faible. Elle est exactement limitée à la taille des oligopeptides traités, multipliée par 3, plus un certain nombre d'indices de boucles. StackPrt contient un générateur de programme pour écrire automatiquement le code source optimisé pour une tâche de traduction inverse, en fonction de la taille de l'oligopeptide. La complexité de l'algorithme StackPrt est limitée à la dégénérescence totale de l'oligopeptide traité.

Soit *p* un oligopeptide de longueur *pl* acides aminés. Chaque acide aminé *i* de *p* peut être codé par *NC(i)* codons correspondants. La dégénérescence totale *D* de cet oligopeptide est donnée par:

$$D = \prod_{i=0}^{pl-1} NC(i)$$

L'espace disque nécessaire *RDS* (« Required Disk Space ») pour stocker tous les oligonucléotides obtenus est calculé en utilisant la formule suivante:

$$RDS = [D * (pl * 3 + 1)] \text{ Octets}$$

Un rapport technique complet, y compris le code source C pour l'algorithme StackPrt est disponible dans (Hill, 2006).

Ci-après, nous montrons le fonctionnement de l'algorithme StackPrt avec un exemple d'un oligopeptide court constitué de trois acides aminés: «*ASR*».

Pour l'acide aminé « *A* » (« Alanine »), nous avons 4 codons potentiels: *GCT, GCC, GCA* et *GCG*. Pour l'acide aminé « *S* » (« Sérine »), nous avons 6 codons potentiels: *TCT, TCC, TCA, TCG, AGT* et *AGC*. Enfin pour l'acide aminé « *R* » (« Arginine »), nous avons 6 codons: *CGT, CGC, CGA, CGG, AGA* et *AGG*.

Les valeurs des variables suivantes sont alors définies pour l'oligopeptide « *ASR* »:

$pl = 3$, $NC(0) = 4$, $NC(1) = 6$ et $NC(2) = 6$.

La dégénérescence *D* et la mémoire de stockage requise *RDS* de l'oligopeptide «*ASR*» sont calculées par StackPrt comme suit:

$D = NC(0) * NC(1) * NC(2) = 4 * 6 * 6 = 144$ oligonucléotides,

$RDS = D * (pl * 3 + 1) = 144 * (3 * 3 + 1) = 1{,}41$ Ko.

Pour réaliser la traduction inverse de l'oligopeptide « *ASR* », StackPrt utilisera trois boucles imbriquées:

(i) Empiler le 1er codon de l'acide aminé « A » dans la pile: Pile = « GCT ».

(ii) Empiler le 1er codon de l'acide aminé « *S* » dans la pile: Pile = « *GCTTCT* ».

(iii) Empiler le 1er codon de l'acide aminé « *R* » dans la pile: Pile = « *GCTTCTCGT* ».

Au niveau de la dernière boucle, StackPrt fournit le 1er oligonucléotide obtenu (« *GCTTCTCGT* ») puis retire le dernier codon de la pile (Pile = « *GCTTCT* »). Ensuite, il empile le prochain codon de l'acide aminé « *R* » (Pile = « *GCTTCTCGC* ») et fournit le second oligonucléotide obtenu (« *GCTTCTCGC* »). StackPrt traite de la même manière les autres codons de l'acide aminé « *R* ». Une fois le dernier codon de « *R* » est traité, StackPrt retire les deux derniers codons de la pile (Pile = « *GCT* ») et empile le deuxième codon de l'acide aminé « *S* » (Pile = « *GCTTCC* ») et le premier codon de l'acide aminé « *R* » (Pile

= « *GCTTCC* »). Un autre oligonucléotide est alors fourni (« *GCTTCC* »). StackPrt achève ensuite le reste de la traduction inverse complète de cet oligopeptide de la même manière.

Dans le cas présenté, chaque codon du premier acide aminé « *A* » n'est écrit qu'une seule fois dans la pile pour l'ensemble des 144 combinaisons possibles. Une énumération complète classique comme celle utilisée dans DegenRev (Missaoui et al., 2007) aurait nécessité 36 écritures de ce même codon. La complexité de l'algorithme StackPrt est limitée à la dégénérescence totale de l'oligopeptide.

StackPrt possède plusieurs avantages tels que:

- ✓ La mémoire nécessaire est très limitée.
- ✓ L'utilisation de la pile limite le nombre d'écritures dans la mémoire.
- ✓ L'efficacité augmente avec la longueur de l'oligopeptide.
- ✓ Les codes sources et binaires sont générés automatiquement en fonction de la longueur des oligopeptides. Les fichiers générés peuvent être réutilisés pour d'autres oligopeptides de la même taille.
- ✓ Comparé à d'autres programmes tels que DegenRev, StackPrt est plus efficace et offre de meilleures performances en temps de calcul (Missaoui et al., 2008).

4.2. Approche proposée

L'utilisation de la traduction inverse complète pour la sélection de sondes pour biopuces à ADN nécessite de traiter jusqu'à plusieurs milliers d'oligopeptides simultanément. Les oligonucléotides produits doivent également répondre à certains critères essentiels de sélection de sondes telles que l'uniformité et la sensibilité (Dugat-Bony et al., 2012b). Dans ce contexte, la traduction inverse complète devient une tâche fastidieuse. C'est pourquoi nous proposons un nouvel algorithme parallèle pour la traduction inverse complète d'un grand nombre d'oligopeptides simultanément, en se basant sur la méthode StackPrt (Hill, 2006). Notre algorithme filtre les oligonucléotides produits pour ne garder que ceux de haute qualité et qui sont bien adaptés à la tâche de sélection de sondes pour biopuces FGAs.

Nous avons implémenté notre méthode dans un programme appelé « P-MetaStackPrt ». Il est développé en langage C++ sous Linux CentOS 5.4. P-MetaStackPrt génère les codes sources nécessaires pour la parallélisation d'une traduction inverse

complète en fonction de l'architecture de calcul choisie par l'utilisateur, le nombre de processeurs et d'autres paramètres d'entrée tels que la taille des oligopeptides traités. Les codes sources sont ensuite compilés et la tâche de traduction inverse est exécutée. Cette méta-programmation a été réalisée en utilisant une approche d'ingénierie dirigée par les modèles (figure 49).

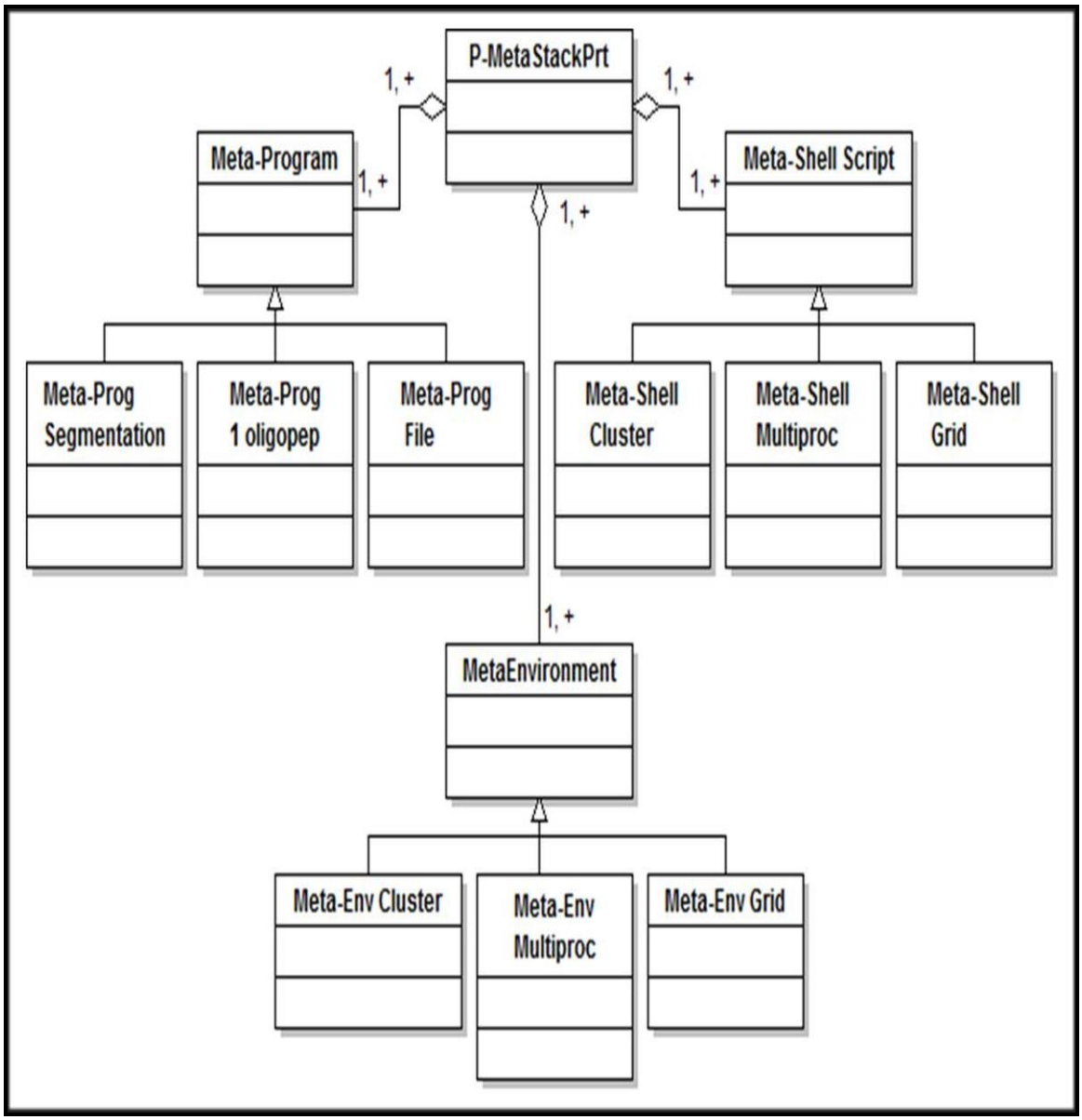

Figure 49. Méta modèle UML de l'algorithme parallèle proposé pour la traduction inverse complète d'un ou plusieurs oligopeptides.

4.3. Adaptation de l'algorithme au problème de sélection de sondes pour biopuces fonctionnelles FGAs

Afin de faciliter l'utilisation de notre programme par un logiciel de sélection de sondes et d'améliorer la qualité des résultats, tous les oligonucléotides produits sont examinés avant de les enregistrer, pour éliminer ceux qui ne sont pas adaptés aux biopuces à ADN. Nous ne gardons que les oligonucléotides qui répondent aux critères suivants:

- ✓ Le pourcentage des nucléotides G+C est compris entre 40 et 60% de la longueur de l'oligonucléotide.
- ✓ t <= Tm <= T, où Tm est la température de fusion de l'oligonucléotide calculée en utilisant la méthode thermodynamique du plus proche voisin, T et t sont respectivement la valeur maximale et minimale autorisée de Tm. Les valeurs de T et t sont définies par l'utilisateur au début de l'exécution de notre programme.
- ✓ L'oligonucléotide ne contient aucun homopolymère de longueur supérieure à 4 nucléotides.

Le filtrage des oligonucléotides générés par la traduction inverse complète des oligopeptides permet de réaliser deux fonctionnalités principales:

- ✓ Réduire considérablement l'espace disque requis pour l'enregistrement des résultats. En effet, un nombre important d'oligonucléotides produits par traduction inverse complète ne sont pas retenus car ils ne sont pas adaptés à la production de sondes pour des biopuces à ADN.
- ✓ Réaliser une étape de prétraitement pour la tâche de sélection de sondes pour biopuces fonctionnelles qui est également particulièrement complexes et requiert souvent un temps de calcul important.

4.4. Stratégie de parallélisation

Notre programme permet de réaliser la traduction inverse complète de manière séquentielle ou parallèle. En effet, si l'utilisateur a besoin de traiter un seul oligopeptide, l'approche séquentielle doit être satisfaisante et nous effectuons la traduction inverse de cet oligopeptide uniquement sur PC. Cependant, le traitement d'un grand nombre d'oligopeptides nécessite beaucoup plus de temps de calcul. Dans ce cas, notre logiciel permet d'utiliser un multiprocesseur, un cluster ou une grille de calcul pour les tâches les plus lourdes.

Si l'utilisateur souhaite traiter plusieurs oligopeptides simultanément, il doit fournir en entrée un fichier contenant tous les oligopeptides à traiter. Afin de distribuer le calcul sur tous les processeurs, notre programme utilise une approche d'équilibrage de charge qui se base sur la dégénérescence de tous les oligopeptides d'entrée. D'abord, la dégénérescence de chaque oligopeptide et la dégénérescence totale cumulée sont calculées. Ensuite, la valeur de la dégénérescence moyenne par processeur, notée *Dm*, est calculée. Puis, la dégénérescence de tous les oligopeptides est vérifiée: Si la dégénérescence moyenne *Dm* est inférieure à la dégénérescence d'un oligopeptide de longueur *pl*, ce dernier est découpé et remplacé par 2 ou 3 oligopeptides de la même longueur *pl* mais avec une dégénérescence inférieure. La somme des dégénérescences de ces nouveaux oligopeptides créés est égale à la dégénérescence de l'oligopeptide remplacé. Ensemble, ils doivent également produire exactement les mêmes oligonucléotides qui peuvent être produits par l'oligopeptide remplacé. Cela permet d'améliorer l'équilibrage de charge. Un exemple de découpage d'oligopeptides est donné par la figure 50. Enfin, les oligopeptides sont classés par ordre décroissant sur la base de leurs dégénérescences et un algorithme de type « Worst fit » (Johnson, 1974) est employé pour distribuer tous les oligopeptides d'entrée sur *nproc* fichiers (*nproc* est le nombre de processeurs à utiliser, ce nombre est défini par l'utilisateur au début de l'exécution de notre programme). La distribution se fait en fonction de la valeur de la dégénérescence de chacun des oligopeptides. En effet, *nproc* fichiers vides sont initialement créés. Ces fichiers peuvent contenir des oligopeptides ayant un total de dégénérescence qui ne dépasse pas la valeur de la dégénérescence moyenne *Dm*. Notre logiciel remplit ensuite ces fichiers. Pour chaque oligopeptide, il vérifie tous les fichiers et sélectionne le fichier possédant le plus grand bloc libre possible dans lequel cet oligopeptide peut être sauvegardé et ce en fonction de sa dégénérescence. L'oligopeptide sera alors ajouté à ce fichier. L'objectif est d'éviter de créer des petits blocs inutilisables, en faisant le reste aussi grand que possible afin de le rendre suffisant pour contenir d'autres oligopeptides. Les fichiers créés doivent avoir à peu près la même valeur de dégénérescence cumulée de tous les oligopeptides qui y sont stockés. Chaque fichier sera ensuite exécuté sur un processeur (un cœur de calcul physique).

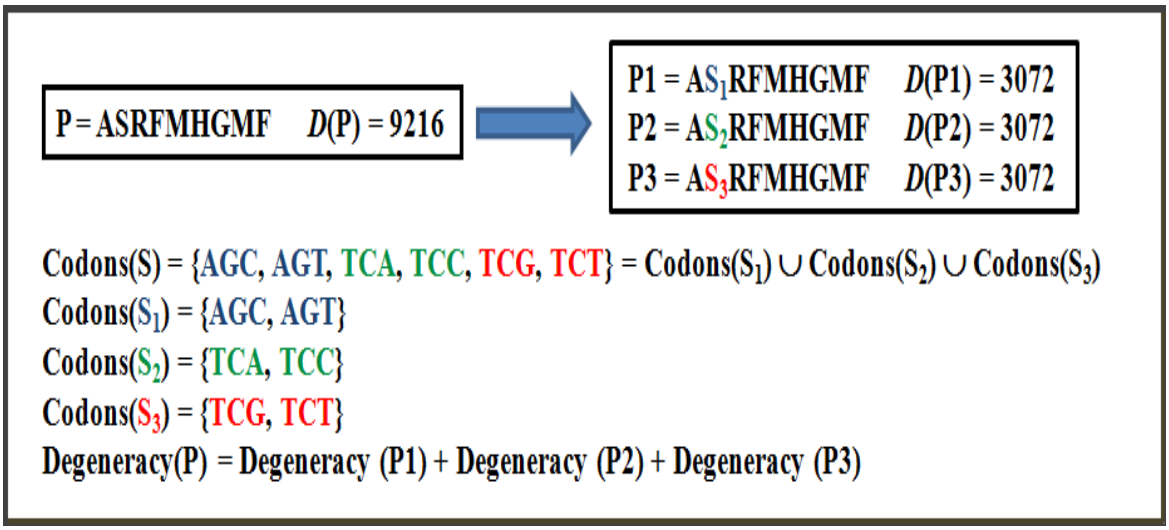

Figure 50. Exemple de découpage d'un oligopeptide pour l'amélioration de l'équilibrage de charge pour une tâche de traduction inverse complète.

Notre logiciel comprend un générateur de programmes qui écrit automatiquement les codes sources nécessaires pour effectuer une tâche de traduction inverse complète sur l'architecture souhaitée. En effet, l'utilisateur sélectionne l'architecture qu'il veut utiliser (un multiprocesseur, un cluster ou une grille de calcul), le nombre de processeurs à utiliser sur cette architecture et la taille des oligopeptides à traiter. Selon ces informations, P-MetaStackPrt génère 2 types de codes sources:

- ✓ Les programmes C++ qui seront utilisés pour paralléliser et réaliser la tâche de traduction inverse complète de l'ensemble des oligopeptides donnés par l'utilisateur.
- ✓ Les scripts shell nécessaires à l'exécution des programmes C++ sur l'architecture désirée, écrits en fonction du nombre de processeurs spécifié par l'utilisateur. Si une grille de calcul est utilisée, outre les fichiers de configuration élémentaire nécessaires pour soumettre des travaux sur la grille (fichiers JDL « Job Description Language » et fichiers shell exécutables contenant les tâches élémentaires qui seront exécutées sur la grille), des scripts supplémentaires sont également générés. Ces scripts servent à surveiller les différents jobs afin d'améliorer la fiabilité de notre logiciel sur la grille. Ils permettent la gestion de l'état des jobs, le transfert des fichiers et la re-soumission automatique des jobs en cas d'échec ou de perte.

Le choix de l'architecture peut être réalisé sur la base de la disponibilité des infrastructures, mais aussi sur la base de la nature de l'application à exécuter. En effet, lorsque la tâche de traduction inverse complète ne requiert pas une grande quantité d'espace disque ou un grand nombre de cœurs de calcul, l'utilisateur peut utiliser un multiprocesseur et les résultats finaux peuvent être complètement stockés sur les disques du multiprocesseur. Cependant, si nous traitons un très grand nombre d'oligopeptides qui nécessitent un temps de calcul considérable et peuvent générer de grandes quantités de données, l'utilisation d'un cluster de calcul est alors le meilleur choix. En fait, pour améliorer les performances de notre programme lors de l'utilisation d'un cluster, nous distribuons toutes les parties du traitement, obtenues par la méthode d'équilibrage de charge, sur plusieurs nœuds du cluster. Chaque processus exécute alors la traduction inverse complète d'une partie des oligopeptides d'entrée et stocke localement sa part des résultats.

En outre, l'utilisation d'une grille de calcul n'est pas intéressante si nous avons seulement un petit groupe d'oligopeptides en entrée ou si la traduction inverse nécessite une grande quantité d'espace disque. Cela est dû au temps nécessaire pour la récupération des résultats et à la limite des capacités de stockage disponibles dans la plupart des éléments de calcul CEs (« Computing Elements ») des grilles. Cependant, si nous traitons plusieurs milliers d'oligopeptides avec une dégénérescence raisonnable, l'utilisation du calcul sur grille est bien adaptée à ce genre d'application. En fait, lorsque notre programme est utilisé simultanément avec un logiciel de sélection de sondes pour biopuces à ADN fonctionnelles, il n'est pas nécessaire de conserver les oligonucléotides produits. Ces derniers peuvent être directement traités un par un pour sélectionner des sondes oligonucléotidiques. Dans ce cas, il faut un temps de calcul très important pour traiter le très grand nombre d'oligopeptides, mais la quantité de résultats retenue n'est souvent pas très grande en raison des critères de sélection de sondes supplémentaires qui sont très stricts telles que la spécificité des sondes (la recherche d'hybridations croisées potentielles contre une base de données de séquences). Ces critères doivent être vérifiés dans la dernière étape de détermination des sondes oligonucléotidiques dans l'outil de sélection de sondes utilisé. Dans ce contexte, l'utilisation d'une grille de calcul est parfaitement adaptée à ce type d'application.

5. Résultats

Nous avons testé notre approche en utilisant des ensembles de données biologiques simulés et réelles. Nous avons utilisé deux jeux de données réels. Le premier est généré à partir d'un gène codant pour une enzyme naphtalène di-oxygénase impliquée dans la dégradation du naphtalène qui a été isolé à partir de la bactérie *Pseudomonas* « sp. LZT5 ». La séquence protéique référencée dans la base de données UniProt/TrEMBL sous l'identifiant (« accession number ») « Q3LTH2 » a été extraite puis utilisée pour générer tous les oligopeptides de 11 acides aminés. Ces oligopeptides génèrent des oligonucléotides de 33-mers. Au total, nous avons obtenu 174 oligopeptides que nous avons sauvegardés dans un fichier prêt à être testé par notre programme. Pour le reste de ce document, nous appelons ce fichier « 174oligopep11 ». Le deuxième ensemble de données est impliqué dans le métabolisme du naphtalène et est composé de 12 307 oligopeptides de longueur 11 acides aminés. Pour le reste de ce document, nous appelons ce fichier « 12307oligopep11 ».

À notre connaissance, aucun logiciel n'est actuellement disponible pour effectuer la traduction inverse complète des oligopeptides pour biopuces fonctionnelles, avec filtrage des oligonucléotides produits. Par conséquent, à part une comparaison avec le programme séquentiel original StackPrt (Hill, 2006) qui propose un meilleur algorithme que celui de DegenRev (Missaoui et al., 2007), nous ne sommes pas en mesure de proposer plus de comparaisons.

Les tests expérimentaux montrent d'une part la performance de notre méthode de parallélisation et d'autre part l'amélioration obtenue en termes de gain d'espace disque en filtrant les oligonucléotides générés. Dans cette section, nous présentons tous les résultats obtenus par notre logiciel.

5.1. Performance de la méthode d'équilibrage de charge utilisée

Pour distribuer les oligopeptides à traiter équitablement sur tous les processeurs utilisés, nous avons développé une méthode d'équilibrage de charge basée sur la dégénérescence des oligopeptides. Comme décrit dans la section 4.4, notre méthode vérifie d'abord tous les oligopeptides donnés par l'utilisateur dans le fichier d'entrée et divise ceux qui possèdent une valeur de dégénérescence supérieure à la valeur de la dégénérescence moyenne, en se basant sur le code UIB/UIPAC des nucléotides dégénérés.

Les oligopeptides obtenus sont ensuite triés par dégénérescence et distribués, en commençant par les plus grands en termes de dégénérescence, sur un nombre donné de sous-fichiers. Chaque sous-fichier créé contient les oligopeptides qui seront traités sur l'un des processeurs utilisés. Pour tester l'efficacité de notre méthode, nous l'avons comparée à deux autres méthodes basées respectivement sur une technique « First Fit » et une technique « Best Fit » (Johnson, 1974). Les tests de comparaison ont été réalisés sur le jeu de données réelles « 174oligopep11 » en utilisant 32 cœurs de calcul sur un SMP de 8 Quad Core AMD Opteron à 2.3 GHz sous Linux. Les résultats obtenus (figure 51) montrent que notre méthode est plus efficace que les deux autres. Grâce à notre méthode d'équilibrage de charge, tous les ensembles d'oligopeptides traités chacun sur un processeur, ont presque la même valeur de dégénérescence totale, qui est très proche de la dégénérescence moyenne par processeur.

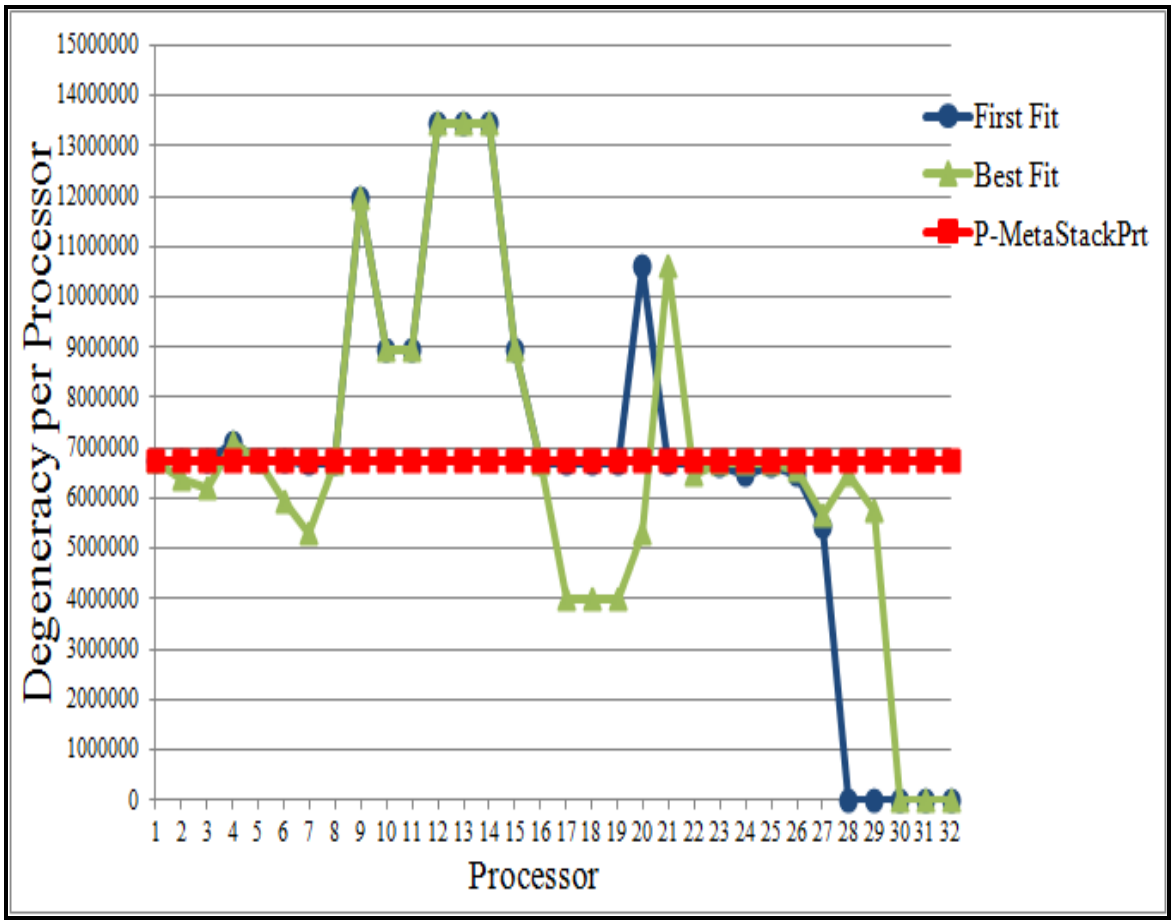

Figure 51. Comparaison de notre méthode d'équilibrage de charge avec 2 autres méthodes basées sur des algorithmes « First Fit » et « Best Fit », en utilisant 32 cœurs de calcul.

5.2. Résultats de l'étape de filtrage des oligonucléotides produits par traduction inverse complète

Pour que les oligonucléotides soient efficaces et bien adaptés aux biopuces fonctionnelles, ils doivent répondre à certains critères de sélection de sondes qui contribuent à la réussite de l'expérience d'hybridation des biopuces. Pour cela, nous avons adapté notre algorithme au problème de développement de biopuces, en ajoutant une étape de filtrage des oligonucléotides générés à partir des séquences d'acides aminés traitées. Ce filtrage se base sur des critères habituels pour la détermination de sondes. L'objectif est de conserver seulement les oligonucléotides qui sont bien adaptés aux expériences de biopuces. Cette étape permet d'améliorer la qualité des résultats obtenus. Elle permet également de réduire considérablement la quantité de données générées par traduction inverse complète et de réduire alors l'espace disque nécessaire pour les sauvegarder. Pour tester notre approche de filtrage d'oligopeptides, nous avons divisé chacun des deux ensembles de données «174oligopep11 » et « 12307oligopep11 » en 10 sous-fichiers en utilisant notre méthode d'équilibrage de charge. Ensuite, nous avons effectué une traduction inverse complète des oligopeptides que nous avions enregistrés dans chacun de ces sous-fichiers, à l'aide de notre logiciel et aussi en utilisant le programme StackPrt. Les résultats obtenus sont illustrés dans les figures 52 et 53 pour respectivement les jeux de données « 174oligopep11 » et « 12307oligopep11 ». Ces figures montrent qu'en utilisant notre approche, la quantité de données conservée après la traduction inverse complète est considérablement réduite. Le gain en terme d'espace disque est d'environ 40% en moyenne pour chacun des deux ensembles de données réelles testés.

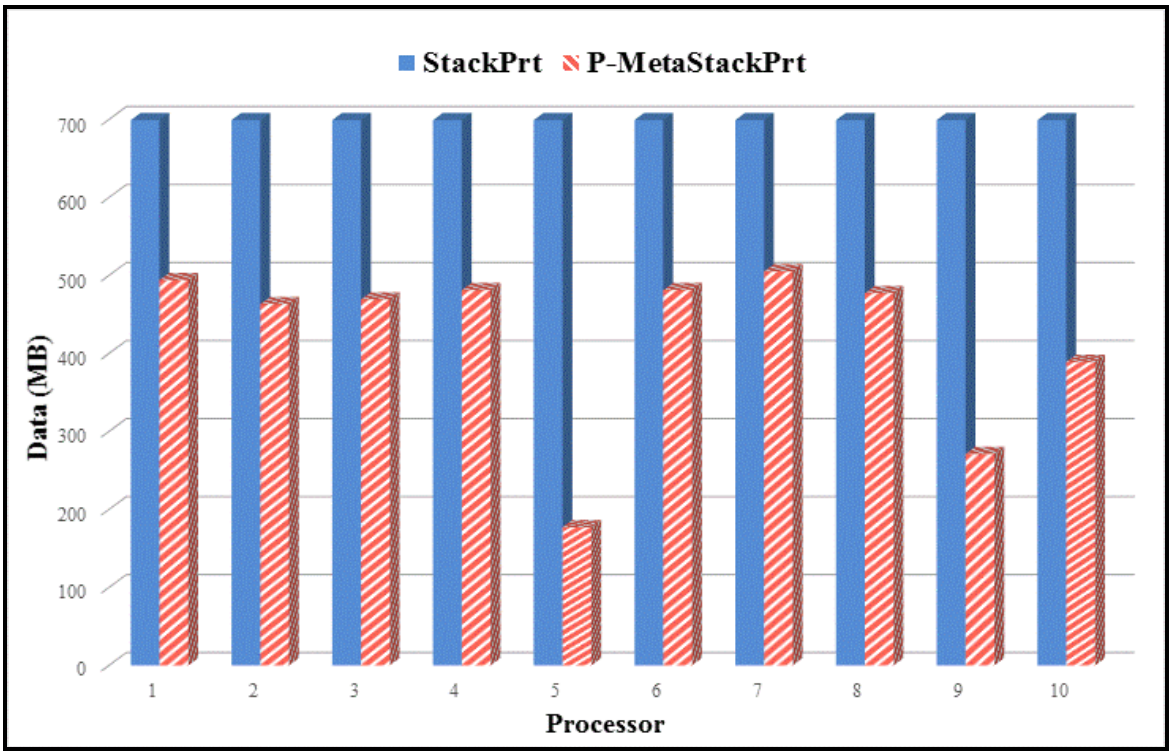

Figure 52. Réduction de la quantité de données produites par traduction inverse complète du fichier « 174oligopep11 » avec filtrage (comparaison avec StackPrt).

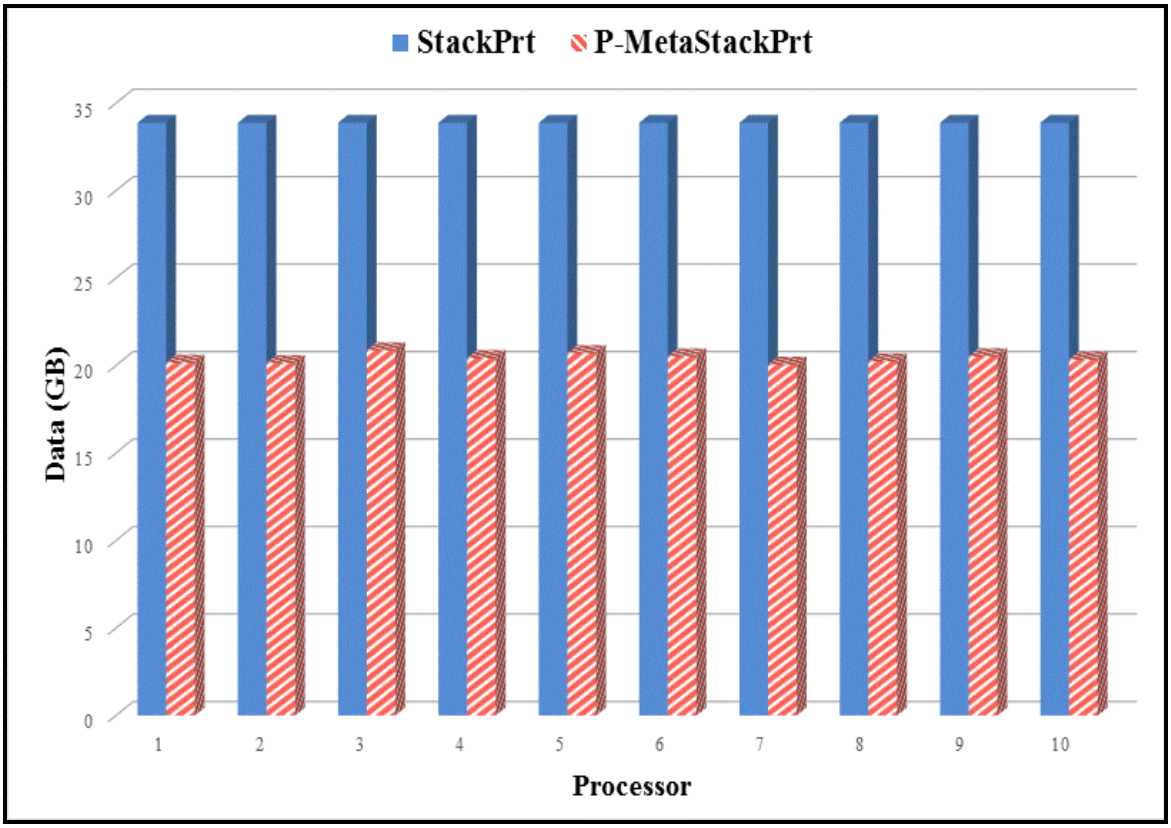

Figure 53. Réduction de la quantité de données produites par traduction inverse complète du fichier « 12307oligopep11 » avec filtrage (comparaison avec StackPrt).

5.3. Résultats de parallélisation de P-MetaStackPrt

Pour effectuer une traduction inverse complète d'un seul oligopeptide, P-MetaStackPrt génère automatiquement un code source séquentiel pour effectuer cette tâche sur un seul processeur. Cependant, la traduction inverse complète de larges ensembles d'oligopeptides peut prendre un temps de calcul considérable. Pour remédier à ce problème, nous avons proposé une méthode de parallélisation efficace qui permet la traduction inverse complète de plusieurs milliers d'oligopeptides simultanément sur différentes architectures de calcul. Quelle que soit l'architecture utilisée, les codes sources générés automatiquement par P-MetaStackPrt sont en langage C++ et sont compilés avec la version 4.1.2 du compilateur g++. Des scripts shell bash sont également générés pour exécuter les programmes C++ sur l'architecture choisie.

Dans cette section, nous montrons la performance de notre implémentation parallèle sur chaque architecture possible.

5.3.1. Tests sur un multiprocesseur

Pour tester le gain réalisé par notre logiciel parallèle lorsqu'on l'exécute l'application sur un multiprocesseur, nous avons utilisé un SMP avec 8 Quad Core AMD Opteron à 2.3 GHz sous Linux. Nous avons réalisé une traduction inverse complète du jeu de données réelles enregistrées dans le fichier « 174oligopep11 », sur 10 cœurs de calcul en utilisant les paramètres suivants:

- ✓ Taille des oligopeptides = 11;
- ✓ Tm minimale = 35;
- ✓ Tm maximale = 70.

Pour calculer le temps nécessaire pour accomplir cette tâche sur un seul cœur de calcul, nous avons utilisé une version modifiée du programme StackPrt (Hill, 2006), dans laquelle nous avions ajouté l'étape de filtrage des oligonucléotides générés. Cette version de StackPrt met environ 43 minutes pour effectuer cette tâche en séquentiel. En utilisant 10 cœurs de calcul, notre programme parallèle P-MetaStackPrt met seulement 4m53s pour accomplir cette tâche. Le speedup est alors d'environ 9x, très proche de 10 le speedup maximal. Les résultats de ce test, avec le temps de calcul sur chaque processeur, sont présentés dans la figure 54.

Nous avons ensuite testé notre programme sur le deuxième jeu de données sauvegardé dans le fichier « 12307oligopep11 ». StackPrt (avec filtrage des oligonucléotides produits) met environ 34 heures pour traiter cet exemple en séquentiel. En utilisant 10 cœurs de calcul, P-MetaStackPrt met seulement 225 minutes (3 heures et 45 minutes) pour traiter le même exemple. Encore une fois, le speedup est d'environ 9x. Les résultats d'exécution, avec le temps de calcul sur chaque processeur, sont présentés dans la figure 55.

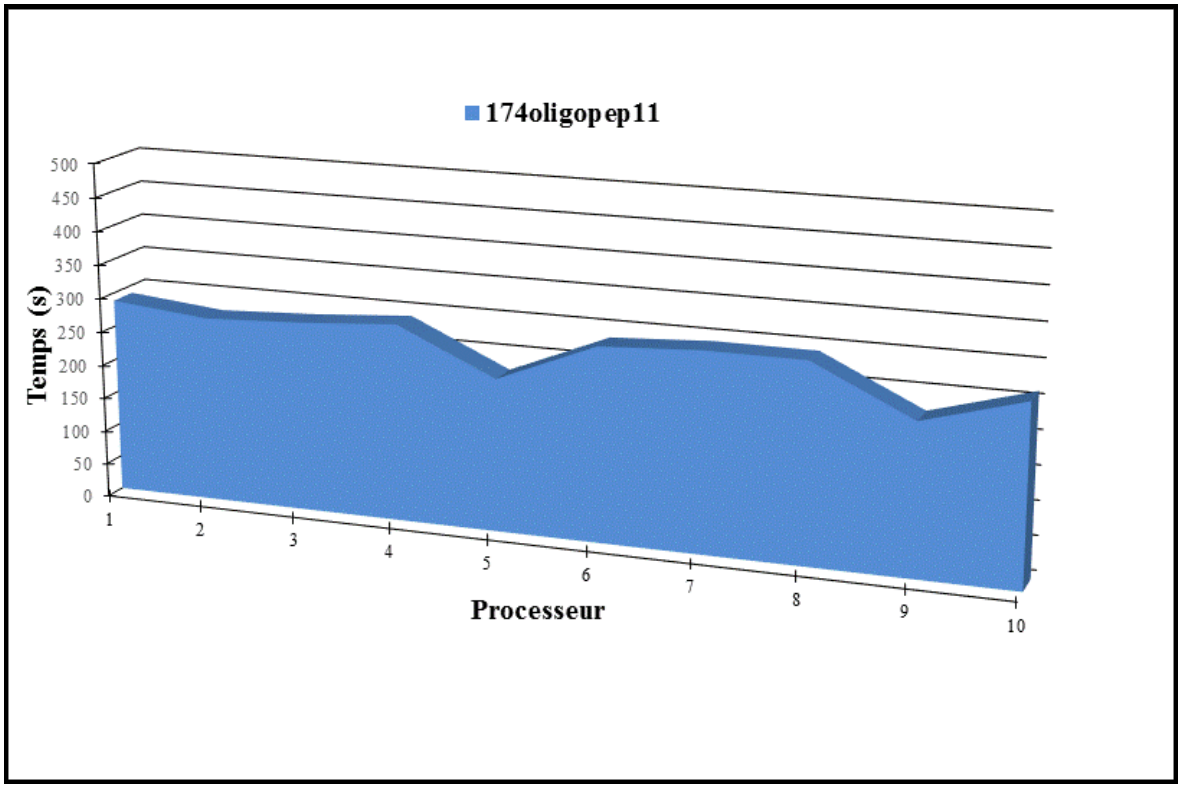

Figure 54. Performance de P-MetaStackPrt pour le traitement du jeu de données biologique « 174oligopep11 » en utilisant un multiprocesseur.

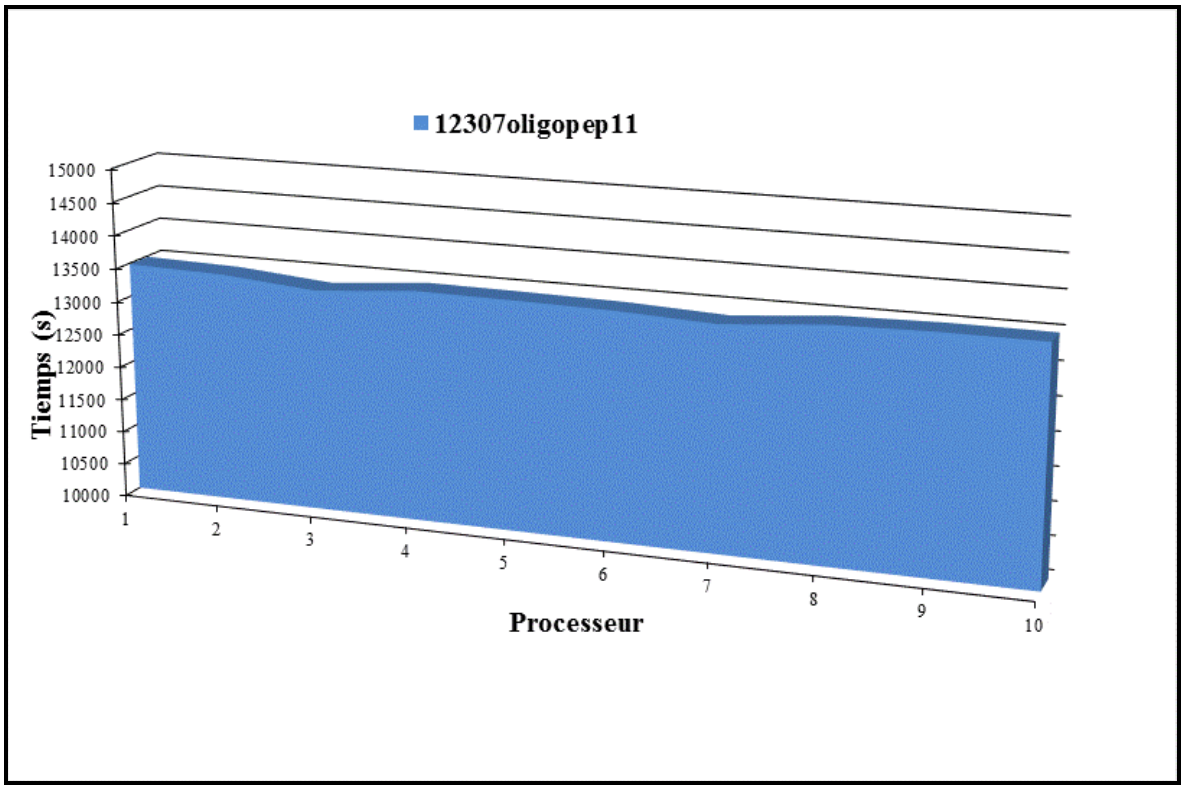

Figure 55. Performance de P-MetaStackPrt pour le traitement du jeu de données biologique « 12307oligopep11 » en utilisant un multiprocesseur.

5.3.2. Tests sur un cluster de calcul

P-MetaStackPrt utilise « PBS » (« Portable Batch System[6] ») pour exécuter et ordonnancer les jobs sur un cluster de calcul. Il écrit automatiquement les fichiers PBS et les scripts shell bash nécessaires pour exécuter et surveiller les jobs.

Pour tester notre logiciel sur un cluster de calcul, nous avons utilisé l'architecture suivante: un système avec 22 nœuds bi-processeurs Quad Core AMD Opteron à 2.1 GHz sous Linux. Nous avons réalisé la traduction inverse complète sur le jeu de données « 12307oligopep11 » en utilisant P-MetaStackPrt avec les paramètres suivants:

- ✓ Taille des oligopeptides = 11;
- ✓ Tm minimale = 35;
- ✓ Tm maximale = 70.

Cette tâche nécessite environ 34 heures de calcul sur un seul cœur de calcul en utilisant StackPrt avec filtrage des oligonucléotides produit. Nous avons exécuté cet

[6] http://www.pbsworks.com/

exemple en utilisant un nombre différent de processeurs. Les résultats illustrés dans la figure 56 montrent clairement la performance de notre logiciel. En effet, nous avons obtenu un speedup d'environ 9x en utilisant 10 processeurs et d'environ 85x en utilisant 100 processeurs, montrant ainsi que notre programme répartit d'une manière performante la charge de calcul.

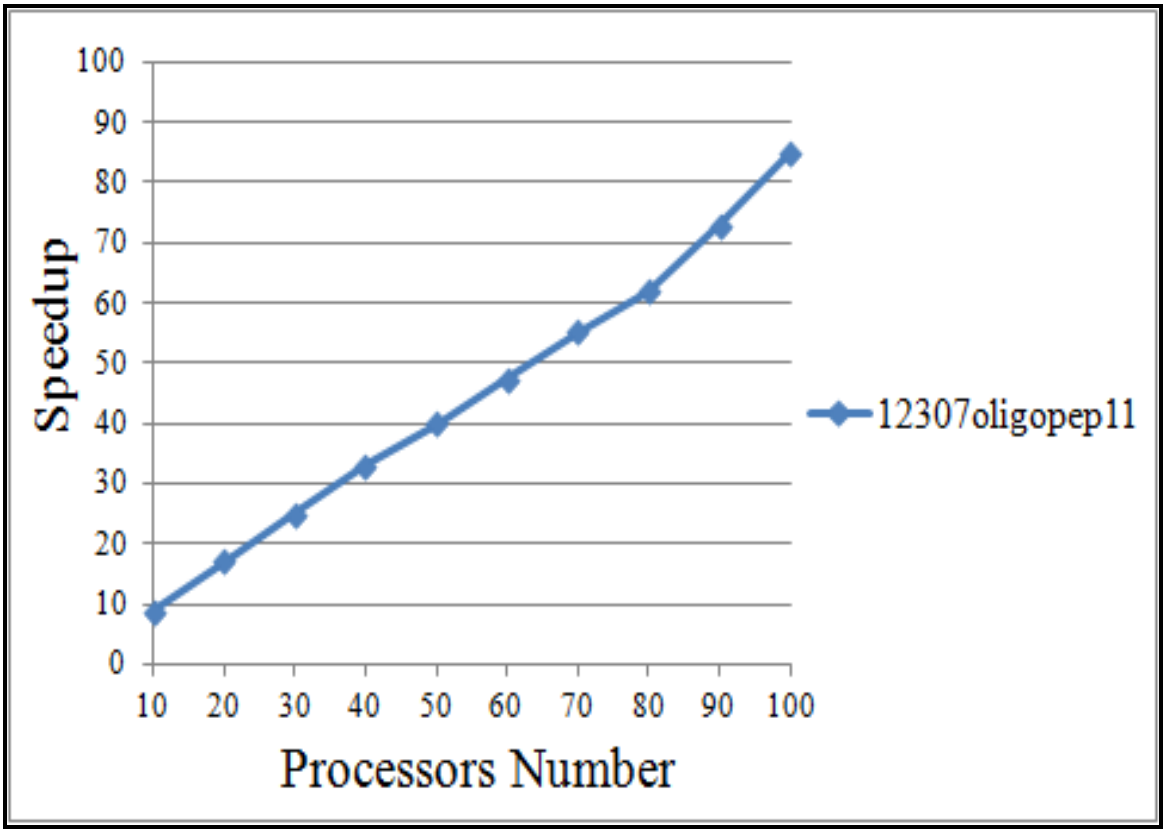

Figure 56. Performance de P-MetaStackPrt en utilisant un cluster de calcul avec différents nombres de processeurs. Le speedup est donné par le quotient entre le temps d'exécution parallèle et celui d'exécution séquentielle.

5.3.3. Tests sur une grille de calcul

Nous avons testé la performance de notre logiciel sur la grille européenne EGI («European Grid Infrastructure[7] »). EGI est une infrastructure de grille multidisciplinaire fournissant plus de 370 000 cœurs CPU et plus de 170 pétaoctets réparties sur plus de 56 pays.

[7] http://www.egi.eu/

L'utilisation d'une grille de calcul pour la traduction inverse complète à grande échelle de plusieurs centaines d'oligopeptides est une approche prometteuse lorsque la tâche de traduction inverse est réalisée simultanément avec celle de sélection de sondes pour biopuces fonctionnelles. En effet, dans ce contexte il n'est pas nécessaire de conserver les oligonucléotides produits par toutes les séquences d'acide aminés traitées. Les oligopeptides obtenus peuvent être directement traités un par un pour sélectionner les sondes oligonucléotidiques. En plus du filtrage des oligonucléotides que notre programme réalise, d'autres critères de sélection stricts doivent être également vérifiés dans un logiciel de détermination de sondes telle que la spécificité (Kane et al., 2000). Ces critères stricts permettront de réduire considérablement le nombre d'oligonucléotides retenus. Pour cela, et afin d'éviter le problème de transfert de gros fichiers sur la grille, nous avons testé notre logiciel sur la grille de calcul européenne sans enregistrer les oligonucléotides générés.

Nous avons sauvegardé seulement le nombre d'oligonucléotides générés par traduction inverse. Les tests ont été effectués sur des jeux de données simulés. Tout d'abord, nous avons utilisé P-MetaStackPrt pour réaliser la traduction inverse complète de 1200 oligopeptides que nous avions enregistrés dans un fichier appelé « 1200oligopep14 ». Les oligopeptides sont tous d'une longueur de 14 acides aminés. Cette tâche nécessite environ 125 jours pour être traitée sur un seul cœur de calcul. Nous avons répété cette tâche 5 fois sur la grille de calcul européenne en soumettant 300 jobs à chaque fois. Comme valeur médiane, nous avons obtenu tous les résultats avec succès après 24 heures seulement (avec la latence liée à l'attente et à la soumission des jobs). Le speedup de notre logiciel est dans ce cas de 125x. L'écart type entre les 5 réplicats exécutés est de 5h. Les résultats sont illustrés dans la figure 57 et dans le tableau 17.

Nous avons ensuite réalisé la traduction inverse complète de 5000 oligopeptides simulés de longueur 16 acides aminés, sur la grille de calcul européenne en utilisant un total de 1000 jobs. Cette tâche nécessite plus de 1 an et 5 mois de calcul pour être exécutée sur 1 seul cœur de calcul. En utilisant la grille EGI, nous avons réalisé la traduction inverse de ce jeu de données en moins de 65 heures (résultat médian de 5 réplicats, avec la latence liée à l'attente et à la soumission des jobs). Les résultats obtenus sont illustrés dans le tableau 17. Dans cet exemple, la performance réalisée est d'environ 200x (résultat médian). L'écart type entre les 5 réplicats exécutés est de 8h.

Les jobs soumis à une grille de calcul peuvent passer des heures à attendre dans les files d'attente. En plus, lors de l'utilisation d'une grille de calcul, la performance de notre

logiciel reste fortement dépendante de la charge de travail effective au même moment sur les ressources de la grille. L'indisponibilité de certaines ressources de la grille, tels que un « CE » (« Computing Element ») ou une « SE » (« Storage Element ») (voir chapitre 3 section 2.6 pour plus de détails sur les éléments d'une grille) peut causer plusieurs problèmes comme la perte ou le blocage des jobs. Cela peut augmenter considérablement le temps de calcul global de notre logiciel qui doit réaliser à chaque fois une re-soumission des jobs perdus ou ceux de ceux qui ont échoué. Pour cela, nous devons utiliser la grille de calcul seulement lorsque nous avons de très grands groupes d'oligopeptides à traiter, sinon le temps passé dans les files d'attente de la grille peut largement dépasser le temps de calcul réel. Pour les petits groupes d'oligopeptides, l'utilisation d'un serveur multiprocesseur ou d'un cluster de calcul est beaucoup plus efficace que l'utilisation d'une grille pour toutes les raisons de latence précédemment présentées.

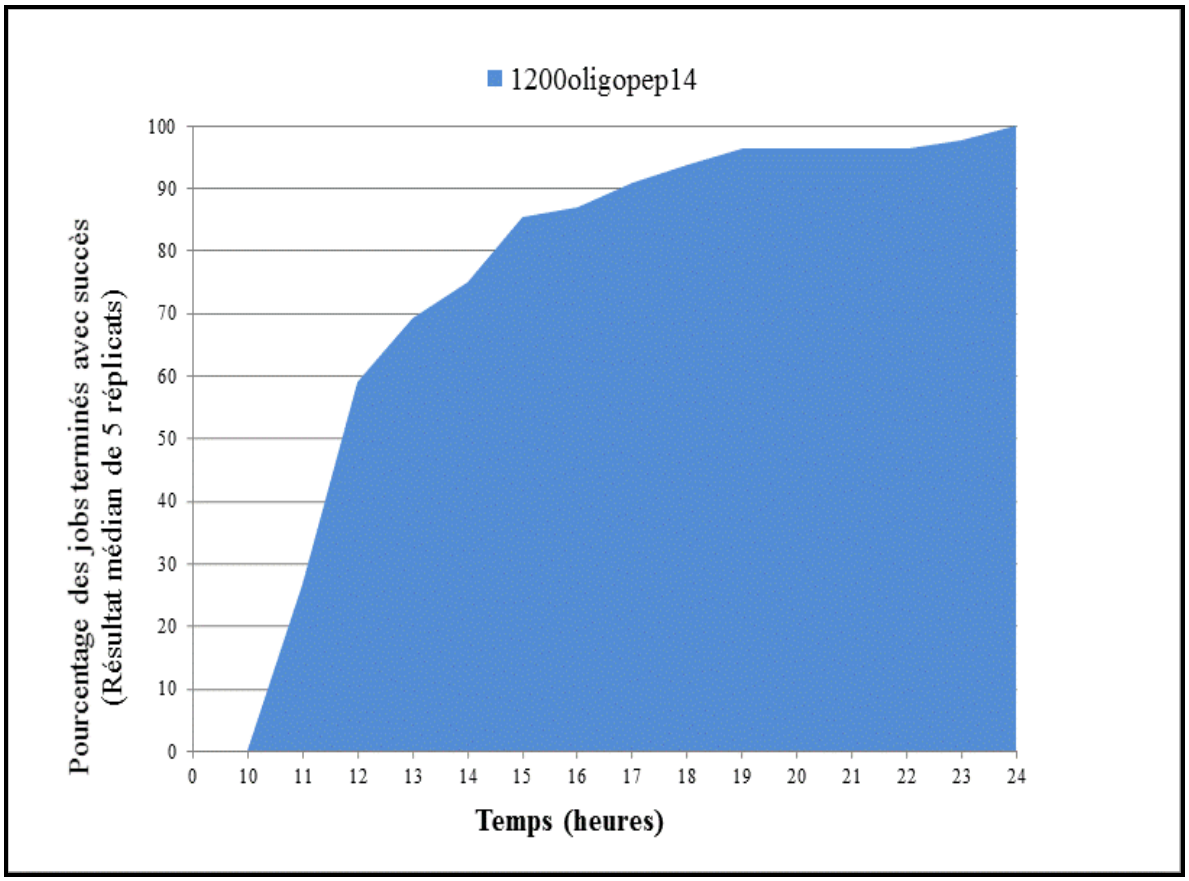

Figure 57. Résultat médian d'exécution de la traduction inverse complète de 1200 oligopeptides de 14 acides aminés, en utilisant 300 jobs sur la grille de calcul EGI.

Tableau 17: Résultats d'exécution de la traduction inverse complète de deux jeux de données simulés sur la grille EGI.

Jeu de données	Temps (heures) (avec la latence liée à l'attente et à la soumission des jobs)				
	Médiane	Moyenne	Min	Max	Ecart type
1200oligopep14	24	27	23	35	5
5000oligopep16	65	70	65	80	8

6. Discussion

P-MetaStackPrt est une approche hybride pour calculer une traduction inverse d'oligopeptides tout en gardant à l'esprit l'objectif final: la sélection de sondes pour biopuces à ADN fonctionnelles. Cette approche hybride combine une distribution de calcul bien équilibrée de la traduction inverse complète, avec un filtrage des oligonucléotides produits par des critères de sélection de sondes.

Notre programme permet de générer toutes les sondes oligonucléotidiques possibles à partir d'un ensemble d'oligopeptides de longueur entre 7 et 17 acides aminés. Les oligonucléotides générés sont filtrés par des critères de sélection de sondes habituels. Ce filtrage permet d'améliorer la qualité des oligonucléotides retenus, dans un contexte de développement de biopuces à ADN. Il permet également l'absorption de la nature exponentielle du problème de traduction inverse complète en réduisant l'espace disque requis pour stocker les résultats. Les tests expérimentaux que nous avons réalisés sur des données biologiques réelles montrent que notre approche a réduit l'espace disque utilisé d'environ 40%. Cependant, les quantités de données retenues après le filtrage des oligonucléotides restent toujours considérables, d'où l'intérêt de l'utilisation des architectures parallèles et distribuées pour le stockage des données. Ces données seront cependant encore réduites lors de l'étape de sélection de sonde, avec la vérification d'autres critères plus stricts telle que la spécificité.

Pour faire face au temps de calcul considérable nécessaire pour le traitement d'un grand nombre d'oligopeptides, nous avons proposé une méthode de parallélisation efficace basée sur la dégénérescence de chaque oligopeptide. Les oligopeptides à traiter sont équitablement répartis sur un nombre donné de fichiers en utilisant une méthode de parallélisation à deux niveaux: intra- et inter-oligopeptides. Nous avons comparé notre

méthode de parallélisation avec 2 autres techniques basées respectivement sur des algorithmes « First fit » et « Best fit ». Nous avons montré que notre approche est plus efficace et permet un partage équitable du traitement des oligopeptides sur tous les processeurs disponibles.

P-MetaStackPrt peut être utilisé pour effectuer la traduction inverse complète de plusieurs milliers d'oligopeptides simultanément sur des architectures parallèles et distribuées. Il propose également une version séquentielle qui peut être utilisée pour réaliser la traduction inverse d'un seul oligopeptide sur un PC. Notre logiciel contient un générateur de programmes qui écrit automatiquement les codes sources nécessaires pour effectuer une tâche de traduction inverse complète sur l'une des 4 architectures proposées: un PC, un multiprocesseur, un cluster ou une grille de calcul. Cette méta-programmation est basée sur une approche d'ingénierie dirigée par les modèles qui réduit considérablement la complexité de notre programme. Nous avons testé les performances de notre logiciel sur des données biologiques réelles et simulées. Tous les résultats expérimentaux obtenus en utilisant un multiprocesseur, un cluster ou une grille de calcul ont montré que la parallélisation de notre logiciel a amélioré de manière significative ses performances avec un speedup jusqu'à 9x pour l'utilisation de 10 cœurs de calcul sur un multiprocesseur et jusqu'à 85x pour l'utilisation de 100 cœurs de calcul sur un cluster. Même lors de la soumission des jobs sur la grille de calcul européenne, nous avons réalisé un speedup de 200x en utilisant 1000 jobs (avec la latence liée à l'attente et à la soumission des jobs).

Quelle que soit l'architecture de calcul utilisée, notre approche est bien adaptée à la sélection de sondes oligonucléotidiques à grande échelle pour biopuces à ADN fonctionnelles. Elle peut être facilement intégrée dans des outils de sélection de sondes. Dans nos travaux futurs, nous allons étudier l'intégration de notre approche dans un outil de sélection de sonde comme «Metabolic Design» (Terrat et al., 2010). L'utilisation de notre approche dans le logiciel de sélection de sondes MetaExploArrays (Jaziri et al., 2012) que nous avons présenté dans le chapitre 5, est aussi envisageable pour permettre la détermination de sondes à partir d'un groupe de séquences nucléotidiques ou protéiques en fonction des besoins de l'utilisateur.

Nous allons également étudier la possibilité d'ajouter de nouvelles architectures pour offrir plus de choix à l'utilisateur. Dans ce contexte, le cloud représente une piste très

intéressante, notamment pour le stockage des grandes quantités de données générées par une traduction inverse complète d'un grand nombre d'oligopeptides simultanément.

7. Conclusion

Dans ce chapitre, nous avons présenté un algorithme distribué efficace pour la traduction inverse complète d'oligopeptides, appliqué à la sélection de sondes pour biopuces à ADN fonctionnelles. Les oligonucléotides générés par notre programme sont filtrés par des critères de sélection de sondes habituels, afin d'améliorer la qualité des oligonucléotides retenus, dans un contexte de développement de biopuces à ADN, et de réduire l'espace disque requis pour stocker les résultats.

Notre programme utilise la méta-programmation et l'ingénierie dirigée par les modèles pour écrire automatiquement les codes sources nécessaires pour la traduction inverse complète d'oligopeptides sur différentes architectures de calcul: un PC, un multiprocesseur, un cluster ou une grille de calcul.

Notre logiciel est bien adapté au problème de sélection de sondes à grande échelle et peut être facilement intégrée dans d'autres logiciels de conception de biopuces à ADN fonctionnelles. Il peut également être intégré dans notre logiciel de sélection de sondes MetaExploArrays (chapitre 5).

Conclusion générale

1. Contributions

L'objectif de cette étude doctorale était de développer de nouveaux algorithmes pour la conception et l'analyse des biopuces à ADN, tout en faisant appel au calcul intensif et aux nouvelles approches du génie logiciel pour améliorer les performances de nos développements. Notons, que malgré l'avènement du séquençage de deuxième génération les approches biopuces ADN restent toujours largement utilisées. Le séquençage de nouvelle génération a permis de produire des masses de données considérables à des coûts réduits. Cependant, l'exploration d'écosystèmes microbiens nécessite des efforts de séquençage considérables qui restent très couteux sans pouvoir forcément obtenir l'information recherchée. Ainsi, de nombreuses approches de séquençage n'utilisent qu'une infime partie des séquences générées pour répondre aux questions posées. Enfin, il demeure difficile de traiter les masses de données de séquences. Aussi, les biopuces ADN par leur simplicité d'utilisation, leur caractère haut débit, les formats de multiplexage d'échantillons et leur facilité d'interprétation sont adaptées pour de nombreuses applications notamment en écologie microbienne.

En effet, la conception de biopuces à ADN est loin d'être une tâche triviale. La complexité de cette tâche est due essentiellement à la difficulté de combiner une multitude de paramètres pour la recherche de sondes spécifiques et sensibles, ainsi qu'au besoin de sélectionner simultanément plusieurs milliers de sondes pour pouvoir exploiter les grandes capacités des nouveaux formats haut débit des biopuces (jusqu'à plusieurs millions de sondes). Les algorithmes de sélection de sondes doivent également traiter une large quantité de données génomiques. Ces données sont actuellement en croissance continue. Cependant, malgré l'explosion du nombre de séquences déposées dans les banques de données internationales, la grande majorité des microorganismes reste inconnue. Dans ce contexte, les biopuces à ADN exploratoires sont capables d'anticiper les variations génétiques et de cibler de nouvelles séquences ou variants de gènes non encore découverts. La conception de ce type de biopuce, avec les formats haut débit actuels, ainsi que l'explosion de l'information dans les bases de données génomiques, est une tâche fastidieuse qui requiert un temps de calcul important (jusqu'à plusieurs jours pour sélectionner des sondes ciblant un seul large groupe de séquences (Missaoui, 2009).

L'utilisation du calcul haute performance et des architectures distribuées telles que les grilles de calcul pour une sélection de sondes à très grande échelle permet des gains de

temps considérables. Pour cela, nous avons développé un logiciel de sélection de sondes régulières et exploratoires pour biopuces phylogénétiques, entièrement déployé sur la grille de calcul européenne EGI, avec des gains de performance considérables (Jaziri et al., 2011, 2014a).

Pour assurer une sélection de sondes de bonne qualité, nous avons proposé une nouvelle approche pour la construction des bases de données de séquences des gènes exprimant la petite sous unité de l'ARN ribosomique, adaptée à la sélection de sondes pour biopuces phylogénétiques. En effet, une base de données de séquences de bonne qualité est la condition indispensable pour la détermination de sondes spécifiques. Nous avons ainsi développé une stratégie pour créer une base de données de séquences de l'ARNr 16S au niveau du genre. Cette base contient plus de 66000 séquences représentant un total de 2069 genres procaryotes. Cette base a été le support de la détermination de sondes utilisant nos nouveaux algorithmes.

Ainsi, nous avons construit une base de données exhaustive de sondes ciblant le biomarqueur 16S (Jaziri et al., 2014b). Cette base, que nous avons appelée PhylOPDb, contient 74 003 sondes 25-mers régulières et exploratoires ciblant un total de 2 178 genres procaryotes. Nous avons développé une interface web d'accès à PhylOPDb, qui assure un accès simple aux sondes sélectionnées pouvant être consultées et téléchargées par les biologistes. L'accès est libre et sans inscription préalable sur le site http://g2im.u-clermont1.fr/PhylOPDb/. PhylOPDb pourrait être utilisée pour construire une biopuce phylogénétique procaryote complète, mais elle est également bien adaptée à d'autres outils moléculaires qui utilisent des amorces ou des sondes: PCR, PCR quantitative, FISH et capture de gènes. PhylOPDb est actuellement la base de données procaryotes d'oligonucléotides la plus complète par rapport à tous les ensembles de sondes ciblant l'ARNr 16S, existants et disponibles. En effet, la sélection de sondes est une tâche particulièrement délicate s'agissant de ce type de biomarqueur (proximité des séquences), ce qui demande une expertise particulière qui n'est présente que dans très peu de laboratoires au niveau international. PhylOPDb est de plus la seule base de données de sondes disponible ciblant le biomarqueur 16S avec des sondes régulières et exploratoires.

Nous avons d'autre part montré dans le chapitre 4, que l'utilisation d'une grille de calcul pour la sélection de sondes pour biopuces à ADN exploratoires permet d'augmenter les performances en temps de calcul, surtout pour les larges groupes de séquences. Cependant, la performance de cette approche dépend fortement du nombre et de la taille

des groupes de séquences ciblés. En effet, pour les petits groupes de séquences, le temps d'attente et de transfert des données sur la grille peut largement dépasser le temps de calcul. Pour ce type de calcul, l'utilisation d'un multiprocesseur ou d'un cluster de calcul est plus adaptée que l'utilisation d'une grille. Dans ce contexte, nous avons développé un nouveau logiciel, appelé « MetaExploArrays », pour la sélection de sondes oligonucléotidiques régulières et exploratoires sur un PC, un multiprocesseur, un cluster ou une grille de calcul (Jaziri et al., 2012). MetaExploArrays utilise une approche de méta-programmation et d'ingénierie dirigée par les modèles, afin de générer automatiquement les codes sources nécessaires pour sélectionner des sondes sur l'architecture de calcul souhaitée. Le choix final de l'architecture utilisée revient à l'utilisateur. Cependant, notre programme contient un script pour guider l'utilisateur et l'aider dans sa prise de décision. Il l'aide à choisir la meilleure architecture possible en fonction de la complexité de la tâche de sélection de sondes à réaliser et de la disponibilité des différentes architectures.

Dans MetaExploArrays, nous avons également introduit une nouvelle stratégie de sélection de sondes exploratoires qui consiste à sélectionner des sondes très spécifiques ciblant exclusivement de potentielles nouvelles espèces du genre ciblé.

La stratégie de sélection de sondes de MetaExploArrays est adaptée à différents types de biopuces, à savoir les POAs (« Phylogenetic Oligonucleotide Arrays »), FGAs (« Functional Gene Arrays ») ou WGAs (« whole genome arrays »). Il suffit simplement d'utiliser la bonne base de données de séquences en fonction de la biopuce à développer. Les oligonucléotides sélectionnés par MetaExploArrays sont également, là encore, bien adaptés à d'autres outils moléculaires: PCR, PCR quantitative, FISH et capture de gènes.

Pour analyser les résultats des biopuces développées avec les deux logiciels de sélection de sondes que nous avons développés dans le cadre de cette thèse, nous avons présenté, dans le chapitre 6, un nouveau programme appelé PhylInterpret qui permet de déterminer la composition d'un échantillon hybridé sur une biopuce à ADN. PhylInterpret énumère toutes les combinaisons possibles d'organismes présents dans l'échantillon traité, en utilisant une approche innovante qui se base sur des notions de la logique propositionnelle. En effet, notre programme transforme le problème d'analyse des résultats de biopuces en un ensemble de formules propositionnelles pour créer ainsi une instance du problème de Satisfaisabilité SAT. Il utilise ensuite un solveur SAT puissant pour déterminer tous les modèles possibles de l'instance SAT créée. Ces modèles représentent toutes les solutions possibles à notre problème initial de détermination de la composition de

l'échantillon hybridé. PhylInterpret peut également être utilisé pour analyser des données issues des expériences d'hybridation des biopuces développées avec d'autres logiciels de sélection de sondes spécifiques des groupes d'organismes tels que PhylArray (Militon et al., 2007) ou KASpOD (Parisot et al., 2012).

PhylInterpret utilise également une approche permettant d'estimer une valeur minimale de réponse de chaque sonde de la biopuce analysée, en se basant sur les intensités d'hybridation des contrôles négatifs et leurs positions sur la biopuce. Cette valeur est utilisée pour déterminer le niveau et la nature de réponse des sondes. Elle peut aussi être utilisée pour contrôler la qualité de l'hybridation et mesurer le niveau d'intensité des hybridations non spécifiques sur la biopuce.

Nous avons aussi proposé une approche hybride pour calculer une traduction inverse complète d'oligopeptides, appliquée à la sélection de sondes pour biopuces à ADN fonctionnelles. Nous avons implémenté notre approche dans un programme appelé P-MetaStackPrt (Jaziri et al., 2013). MetaStackPrt combine une distribution de calcul bien équilibrée de la traduction inverse complète, avec un filtrage des oligonucléotides générés par des critères de sélection de sondes habituels. Ce filtrage permet d'améliorer la qualité des oligonucléotides retenus et permet également l'absorption d'une partie de la nature exponentielle du problème de traduction inverse complète d'oligopeptides en minimisant complètement l'espace mémoire nécessaire pendant le calcul et en réduisant l'espace disque requis pour le stockage des oligonucléotides générés. Les tests expérimentaux que nous avons réalisés sur des données biologiques réelles montrent que notre approche a réduit l'espace disque utilisé d'environ 40 %. En outre, afin de faire face au temps de calcul considérable nécessaire pour le traitement d'un grand nombre d'oligopeptides, nous avons proposé une méthode de parallélisation intra- et inter-oligopeptides pour permettre l'exécution de cette tâche sur l'une les architectures de calcul suivantes: multiprocesseurs, clusters ou une grille de calcul. Les codes sources nécessaires pour l'utilisation de ces architectures sont écrits automatiquement. Cette méta-programmation est basée sur une approche d'ingénierie dirigée par les modèles. Tous les tests expérimentaux réalisés sur des données biologiques réelles et simulées ont montré que quelle que soit l'architecture de calcul utilisée, la parallélisation de notre logiciel a amélioré de manière significative ses performances.

2. Perspectives

Plusieurs applications de nos outils sont déjà en cours. En effet, nous validons une biopuce phylogénétique procaryote complète composée de l'ensemble des oligonucléotides de notre base de données de sondes régulières et exploratoires PhylOPDb. Cette biopuce permettra d'étudier la structure des communautés procaryotes quel que soit l'environnement considéré. En outre, pour améliorer la base de données PhylOPDb, nous envisageons de déterminer de nouvelles sondes phylogénétiques pour des écosystèmes spécifiques (sols, systèmes aquatiques, microbiote intestinaux, etc.). En effet, les tests de spécificité des sondes sont généralement réalisés sur des bases de données de séquences généralistes et non pas sur le sous-ensemble approprié de séquences en fonction de l'environnement étudié. Cela est dû principalement à l'absence de bases de données spécialisées pour l'écologie microbienne. En adaptant notre algorithme de construction des bases de données de séquences des gènes exprimant la petite sous unité de l'ARN ribosomique (chapitre 4), nous pourrons construire des bases de données de séquences réduites réservées uniquement aux écosystèmes étudiés. Ces bases de données seront utilisées pour la sélection de sondes encore plus spécifiques dédiées à l'exploration d'écosystèmes particuliers. Elles seront aussi disponibles en accès libre comme les autres ensembles de sondes actuellement disponibles dans PhylOPDb.

La collection de sonde actuelle de PhylOPDb pourra aussi être enrichie par de nouvelles sondes ciblant les séquences de l'ARNr 18S et permettant d'étudier par exemple les espèces fongiques ou d'autres microorganismes eucaryotes. Nous pourrons sélectionner ces sondes à l'aide de notre nouveau logiciel de sélection de sondes MetaExploArrays, en utilisant toutes les architectures de calcul disponibles: PC, multiprocesseur, cluster et grille de calcul. Ici, avec l'évolution des architectures de calcul distribuées, nous pourrons également étendre les fonctionnalités de MetaExploArrays par l'ajout de la possibilité d'utiliser d'autres architectures tel que le Cloud par exemple.

Les biopuces à ADN qui seront développées avec tous ces ensembles de sondes que nous envisageons de sélectionner, pourront être analysées avec PhylInterpret, notre logiciel d'analyse des données issues des biopuces. Ce logiciel contient une approche de détermination de la réponse des sondes, basée sur les intensités d'hybridation des contrôles négatifs. Cependant, cette approche dépend fortement de la position et de la qualité des contrôles négatifs utilisés. Ces contrôles doivent être de bonne qualité (ne s'hybrident pas

avec des séquences de l'échantillon hybridé) et dispersés de façon régulière sur toute la surface de la biopuce. Dans ce contexte, nous envisageons de développer un nouvel algorithme pour la sélection de sondes de contrôle négatives pour les biopuces à ADN, en fonction de l'échantillon à hybrider. Nous souhaitons également développer un algorithme permettant de positionner tous les contrôles utilisés, d'une manière régulière et équitable, sur la surface entière de la biopuce. Nous envisageons également de paralléliser notre algorithme de calcul de la valeur minimale de réponse d'une sonde, en utilisant des cartes accélératrices pour calculs parallèles. En effet, les processeurs graphiques GPGPUs (« General-purpose computing on graphics processing units ») ou les nouveaux coprocesseurs Xeon Phi d'Intel[1] peuvent être une alternative prometteuse pour réduire la complexité et le temps de calcul de notre algorithme, spécialement quand il s'agit d'analyser simultanément plusieurs biopuces de haute densité.

Trois de nos contributions développées dans le cadre de cette thèse utilisent les grilles de calcul pour permettre de réaliser des calculs à haute échelle. Cependant, bien que nos logiciels contiennent des scripts de supervision et de re-soumission des jobs en cas d'échec, leur performance reste fortement dépendante de la charge de travail effective au même moment sur les ressources de la grille. L'indisponibilité de certaines ressources telles qu'un « CE » (« Computing Element ») ou un « SE » (« Storage Element ») peut causer plusieurs problèmes comme la perte ou le blocage des jobs. Ici, l'utilisation d'un outil informatique pour la gestion des tâches de calcul et des données distribuées tel que DIRAC[2] (« Distributed Infrastructure with Remote Agent Control ») (Tsaregorodtsev et al., 2003; Casajus et al., 2012) est envisageable pour améliorer la fiabilité de nos logiciels.

Pour permettre aux communautés bioinformatiques et biologiques d'utiliser gratuitement nos développements pour la sélection d'ensembles de sondes adaptés à leurs propres travaux ou pour l'analyse de leurs données issues des biopuces à ADN, nous envisageons également, le développement d'une application web regroupant tous les logiciels développés dans le cadre de cette thèse. Cette application constituera avec le site web PhylOPDb une plateforme complète de conception, d'hybridation et d'analyse des biopuces à ADN pour l'écologie microbienne.

[1] http://www.intel.fr/content/www/fr/fr/processors/xeon/xeon-phi-detail.html

[2] http://diracgrid.org/

Bibliographie

Abrahams JP., Van Den Berg M., Van Batenburg E. and Pleij CWA. (1990) Prediction of RNA secondary structure, including pseudoknotting, by computer simulation, *Nucleic Acids Res.*, 18, 3035-3044.

Ahmed, N., Claesson, MJ., O'Sullivan, O., Wang, Q., Nikkilä, J., Marchesi, JR. et al. (2009) Comparative Analysis of Pyrosequencing and a Phylogenetic Microarray for Exploring Microbial Community Structures in the Human Distal Intestine, *PLoS One*, 4, e6669.

Afgan, E., Sathyanarayana, P. and Bangalore, P. (2006) Dynamic Task Distribution in the Grid for BLAST, Proceedings of *IEEE International Conference on Granular Computing, Atlanta, Georgia, USA*, 554-557.

Ahn, J., Yang, L., Paster, BJ., Ganly, I., Morris, L., Pei, Z. et al. (2011) Oral Microbiome Profiles: 16S rRNA Pyrosequencing and Microarray Assay Comparison, *PLoS One*, 6, e22788.

Allender, E., Bauland, M., Immerman, N., Schnoor, H. and Vollme, H. (2009) The complexity of satisfiability problems: Refining Schaefer's theorem, *Journal of Computer and System Sciences*, 75(4), 245–254.

Alm, EW., Oerther, DB., Larsen, N., Stahl, DA. and Raskin, L. (1996) The oligonucleotide probe database, *Appl. Environ. Microbiol.*, 62, 3557–3559

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipmanl, D.J. (1990) Basic Local Alignment Search Tool, *Journal of Molecular Biology*, 215, 403-410.

Amann, R., Krumholz, L. and Stahl, D. (1990) Fluorescent-oligonucleotide probing of whole cells for determinative, phylogenetic, and environmental studies in microbiology, *Journal of bacteriology*, 172, 762-770.

Amann, R., Ludwig, W., and Schleifer, K. (1995) Phylogenetic identification and in situ detection of individual microbial cells without cultivation, *Microbiol Rev.*, 59; 143-169.

Ansorge, WJ. (2009) Next-generation DNA sequencing techniques, *New Biotechnology*, 25, 195-203.

Ashelford, KE., Weightman, AJ. and Fry, JC. (2002) PRIMROSE: a computer program for generating and estimating the phylogenetic range of 16S rRNA oligonucleotide probes and primers in conjunction with the RDP-II database, *Nucleic Acids Res.*, 30, 3481-3489.

Atkinson, C. and Kuhne, T. (2003) Model-driven development: a metamodeling foundation, *IEEE Software*, 20, 36-41.

Azuaje, F. (2001) A computational neural approach to support the discovery of gene function and classes of cancer, *IEEE Trans Biomed Eng.*, 48, 332-339.

Baber, I., Tamby, JP., Manoukis, NC., Sangare, D., Doumbia, S., Traore, SF., Maiga, MS. and Dembele, D. (2011) A python module to normalize microarray data by the quantile adjustment method, *Infect Genet Evol.*, 11(4), 765–768.

Bae, JW. and Park, YH. (2006) Homogeneous versus heterogeneous probes for microbial ecological microarrays, *Trends Biotechnol.*, 24, 318-323.

Bairoch, A. and Boeckmann, B. (1993) The SWISS-PROT protein sequence data bank, recent developments, *Nucleic Acids Res.*, 21, 3093-3096.

Bairoch, A. and Boeckmann, B. (1994) The SWISS-PROT protein sequence data bank: current status, *Nucleic Acids Res.*, 22, 3578-3580.

Bairoch, A. and Apweiler, R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000, *Nucleic Acids Res.*, 28, 45-48.

Baldi, P. and Long, A. (2001) A Bayesian framework for the analysis of microarray expression data: regularized t-test and statistical inferences of gene changes, *Bioinformatics*, 17(6), 509–519.

Bargen, B. and Donnelly, P. (1998) Inside DirectX, *Microsoft Programming Series, Microsoft Press*, 545p.

Barra, V. (2004) modélisation, classification et fusion de données biomédicales. Application à l'imagerie du cerveau et des puces à ADN, *Habilitation à Diriger des Recherches, UBP Clermont-Ferrand, France*, 125 p.

Bayer, MM. and Sinnott, R. (2005) Distributed BLAST in a Grid Computing Context, *In Heidelberg, S.B. (ed), Computational Life Sciences*, 241-252.

Bédrine-Ferran, H., Le Meur, N., Gicquel, I., Le Cunff, M., Soriano, N., Guisle, I., Mottier, S., Monnier, A., Teusan, R., Fergelot, P., Le Gall, JY., Léger, J. and Mosser, J. (2004) Transcriptome variations in human CaCo-2 cells: a model for enterocyte differentiation and its link to iron absorption, *Genomics*, 83(5), 772-89.

Bell, G. (1992) Ultracomputers: A teraflop before its time, *Communications of the ACM*, 35(8), 27–47.

Beltrame, F., Papadimitropoulos, A., Porro, I., Scaglione, S., Schenone, A., Torterolo, L. and Viti, F. (2007) GEMMA - A Grid environment for microarray management and analysis in bone marrow stem cells experiments, *Future Generation Computer Systems*, 23, 382-390.

Ben-Dor, A., Shamir, R. and Yakhini, Z. (1999) Clustering gene expression patterns, *J. Comput Biol.*, 6(3-4), 281-97.

Bers, K., Sniegowski, K., Albers, P., Breugelmans, P., Hendrickx, L., De Mot, R. and Springael, D. (2011) A molecular toolbox to estimate the number and diversity of Variovorax in the environment: application in soils treated with the phenylurea herbicide linuron, *FEMS Microbiol Ecol*, 76, 14-25.

Bézivin, J. and Gerbé, O. (2001) Towards a Precise Definition of the OMG/MDA Framework, *Proceedings of the 16th IEEE international conference on Automated software engineering (ASE 2001), San Diego, USA*, 273-280.

Bezivin, J. (2004) In Search of Basic Principle for Model Driven Engineering, *Novatica Journal, Special Issue, March-April, 2004*, 2, 21-24.

Bezivin, J. (2005) On the unification power of models, *Software and Systems Modeling*, 4, 171-188.

Biere, A., Cimatti, A., Clarke, EM. and Zhu, Y. (1999) Symbolic model checking without bdds, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, 193–207.

Bilban, M., Buehler, LK., Head, S., Desoye, G. and Quaranta, V. (2002) Normalizing DNA microarray data, *Curr Issues Mol Biology*, 2002, 4, 57-64.

Blair, CE., Jeroslow, RG. and Lowe, JK. (1986) Some results and experiments in programming techniques for propositional logic, *Comput. Oper. Res.*, 13, 633–645.

Blanchet, C. and Loomis, C. (2010) La technologie « cloud », *école thématique GRISBI, Roscoff, France, www.grisbio.fr/documents/ecole2010, accessed April 20, 2014*, 11p.

Blaskovic, D. and Barak, I. (2005) Oligo-chip based detection of tick-borne bacteria, FEMS Microbiol Lett, 243, 473-478.

Blazewicz, J., Frohmberg, W., Kierzynka, M. and Wojciechowski, P. (2013) image-MSA — A GPU-based, fast and accurate algorithm for multiple sequence alignment, *Journal of Parallel and Distributed Computing*, 73 (1), 32-41.

Blow, N. (2008) DNA sequencing: generation next-next, *Nature Methods*, 5, 267-274.

Bodrossy, L., Stralis-Pavese, N., Murrell, J.C., Radajewski, S., Weilharter, A., and Sessitsch, A. (2003) Development and validation of a diagnostic microbial microarray for methanotrophs, *Environ Microbiol*, 5, 566-582.

Boeckaert, C., Vlaeminck, B., Fievez, V., Maignien, L., Dijkstra, J. and Boon, N. (2008) Accumulation of trans C18:1 fatty acids in the rumen after dietary algal supplementation is associated with changes in the Butyrivibrio community, *Appl Environ Microbiol*, 74, 6923-6930.

Bolstad, BM., Irizarry, RA., Astrand, M. and Speed, TP. (2003) A comparison of normalization methods for high density oligonucleotide array data based on variance and bias, *Bioinformatics*, 2003, 19(2), 185-93.

Bommarito, S., Peyret, N., and SantaLucia, J., Jr. (2000) Thermodynamic parameters for DNA sequences with dangling ends, *Nucleic Acids Res*, 28, 1929-1934.

Bontemps, C., Golfier, G., Gris-Liebe, C., Carrere, S., Talini, L. and Boivin-Masson, C. (2005) Microarray based detection and typing of the Rhizobium nodulation gene nodC: potential of DNA arrays to diagnose biological functions of interest, *Appl Environ Microbiol*, 71, 8042-8048.

Booch, G., Brown, A., Iyengar, S., Rumbaugh, J. and Selic, B. (2004) An MDA Manifesto, *Business Process Trends/MDA Journal*, 9p.

Bottari, B., Ercolini, D., Gatti, M. and Neviani, E. (2006) Application of FISH technology for microbiological analysis: current state and prospects, *Applied Microbiology and Biotechnology*, 73, 485-494.

Boyce, R., Chilana, P. and Rose, TM. (2009) iCODEHOP: a new interactive program for designing COnsensus-DEgenerate Hybrid Oligonucleotide Primers from multiply aligned protein sequences, *Nucleic Acids Res.*, 37, W222-228.

Breitling, R., Armengaud, P., Amtmann, A. and Herzyk, P. (2004) Rank Products: A simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments, *FEBS Lett.*, 573(1-3), 83-92.

Brodie, EL., Desantis, TZ., Joyner, DC., Baek, SM., Larsen, JT., Andersen, GL., Hazen, TC., Richardson, PM., Herman, DJ., Tokunaga, TK., Wan, JM. and Firestone, MK. (2006) Application of a high-density oligonucleotide microarray approach to study bacterial population dynamics during uranium reduction and reoxidation, *Applied and environmental microbiology*, 72, 6288-6298.

Brodie, E.L., DeSantis, T.Z., Parker, J.P., Zubietta, I.X., Piceno, Y.M. and Andersen, G.L. (2007) Urban aerosols harbor diverse and dynamic bacterial populations, *Proceedings of the National Academy of Sciences of the USA*, 104, 299-304.

Buckner, J., Wilson, J., Seligman, M., Athey, B., Watson, S. and Meng, F. (2010) The gputools package enables GPU computing in R, *Bioinformatics*, 26(1), 134-5.

Butte, A. (2002) The use and analysis of microarray data, *Nat Rev Drug Discov*, 1(12), 951-60.

Candela, M., Consolandi, C., Severgnini, M., Biagi, E., Castiglioni, B., Vitali, B. et al. (2010) High taxonomic level fingerprint of the human intestinal microbiota by ligase detection reaction--universal array approach, *BMC Microbiol*, 10, 116.

Cao, F., Wagner, RA., Wilson, KD., Xie, X., Fu, JD. Drukker, M., Lee, A., Li, RA., Gambhir, SS., Weissman, IL., Robbins, RC. and Wu, JC. (2008) Transcriptional and Functional Profiling of Human Embryonic Stem Cell-Derived Cardiomyocyte, *PLoS ONE*, 3(10), e3474.

Carpenter, A. (2009) cuSVM: a CUDA implementation of support vector classification and regression, *http://patternsonascreen.net/cuSVMDesc.pdf, accessed April 20, 2014*, 9p.

Carvalho, PC., Glória, RV., de Miranda, AB. and Degrave, WM. (2005) Squid – a simple bioinformatics grid, *BMC Bioinformatics*, 6, 197.

Casajus, A., Ciba, K., Fernandez, V., Graciani, R., Hamar, V., Mendez, V. et al. (2012) Status of the dirac project, *J. Phys. Conf. Ser.*, 396(3), doi:10.1088/1742-6596/396/3/032107, 14p.

Castiglioni, B., Rizzi, E., Frosini, A., Sivonen, K., Rajaniemi, P., Rantala, A. et al. (2004) Development of a universal microarray based on the ligation detection reaction and 16S rrna gene polymorphism to target diversity of cyanobacteria, *Applied and environmental microbiology*, 70, 7161-7172.

Caux, J. (2012) Parallélisation et optimisation d'un simulateur de morphogénèse d'organes Application aux éléments du rein, *Thèse de doctorat, Université Blaise Pascal Clermont Ferrand, France*, 267p.

Cetinkaya, D., Verbraeck, A. and Seck, MD. (2011) MDD4MS: A Model Driven Development Framework for Modeling and Simulation, *Proceedings of the 2011 Summer Computer Simulation Conference, The Hague, Netherlands*, 203-211.

Chatziioannou, A., Kanaris, I., Doukas, C., Moulos, P., Kolisis, FN. and Maglogiannis, I. (2011) GRISSOM Platform: Enabling Distributed Processing and Management of

Biological Data Through Fusion of Grid and Web Technologies, *IEEE Transactions on information technology in biomedicine*, 15(1), 83-92.

Chou, C.C., Chen, C.H., Lee, T.T., and Peck, K. (2004) Optimization of probe length and the number of probes per gene for optimal microarray analysis of gene expression, *Nucleic Acids Res*, 32, e99.

Chung, WH., Rhee, SK., Wan, XF., Bae, JW., Quan, ZX. and Park, YH. (2005) Design of long oligonucleotide probes for functional gene detection in a microbial community, *Bioinformatics*, 21, 4092-4100.

Chu, G., Li, J., Narasimhan, B., Tibshirani, R. and Tusher, V., http://www-stat.stanford.edu/~tibs/SAM/sam.pdf, accessed August 20, 2013, "Significance Analysis of Microarrays" Users guide and technical document, 42p.

Claesson, MJ., O'Sullivan, O., Wang, Q., Nikkila, J., Marchesi, JR., Smidt, H., De Vos, WM., Ross, RP. and O'Toole, PW. (2009) Comparative Analysis of Pyrosequencing and a Phylogenetic Microarray for Exploring Microbial Community Structures in the Human Distal Intestine, *Plos One*, 4(8), e6669.

Cleveland, WS. (1979) Robust Locally Weighted Regression and Smoothing Scatterplots, *Journal of the American Statistical Association*, 74(368), 829–836.

Cleveland, WS. (1981) LOWESS: A program for smoothing scatterplots by robust locally weighted regression, *The American Statistician*, 35(1), 54.

Cole, J.R., Wang, Q., Cardenas, E., Fish, J., Chai, B., Farris, R.J. et al. (2009) The Ribosomal Database Project: improved alignments and new tools for rRNA analysis, *Nucleic Acids Res*, 37, D141-145.

Combemale, B. (2008) Ingénierie Dirigée par les Modèles (IDM) État de l'art, *http://hal.archives-ouvertes.fr/docs/00/37/15/65/PDF/mde-stateoftheart.pdf, accessed April 20, 2014*, 19p.

Comfort, J., Hayman, C., Hupfer, C., Pearson, N., Sowell, C., Gordon, D. et al. (2013) Under cloud cover: How leaders are accelerating competitive differentiation, *IBM Center for Applied Insights, IBM Corporation, NY, USA*, 12p.

Cook, SA. (1971) The complexity of theorem-proving procedures, *Proceedings of the 3rd annual ACM synmposium on Theory of computing*, 151–158.

Cook, KL. and Sayler, GS. (2003) Environmental application of array technology: promise, problems and practicalities, *Current opinion in biotechnology*, 14, 311-318.

Cornish-Bowden, A. (1985) Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984, *Nucleic Acid Res.*, 13(9), 3021-3030.

Cottrell, M. and Letrémy, P. (2005) Missing values: processing with the Kohonen algorithm, *Proceedings of XIth International Symposium on Applied Stochastic Models and Data Analysis (ASMDA), Brest, France*, 489-496.

Creignou, N., Khanna, S. and Sudan, M. (2001) Complexity classifications of boolean constraint satisfaction problems, *Society for Industrial and Applied Mathematics, Philadelphia, PA*, DOI: http://dx.doi.org/10.1137/1.978089871854, 106p.

Crescenzi, M. and Giuliani, A. (2001) The main biological determinants of tumor line taxonomy elucidated by a principal component analysis of microarray data, *FEBS Lett.*, 507(1), 114-118.

Cruz-Martinez, K., Suttle, KB., Brodie, EL., Power, ME., Andersen, GL. and Banfield, JF. (2009) Despite strong seasonal responses, soil microbial consortia are more resilient to long-term changes in rainfall than overlying grassland, *ISME J.*, 3, 738-744.

Dabney, AR. and Storey, JD. (2007), Normalization of two-channel microarrays accounting for experimental design and intensity-dependent relationships, *Genome Biol*, 8(3), R44.

DeAngelis, KM., Brodie, EL., DeSantis, TZ., Andersen, GL., Lindow, SE., and Firestone, MK. (2009) Selective progressive response of soil microbial community to wild oat roots, *ISME J.*, 3(2):168-78.

Debnath, R. and Kurita, T. (2010) An evolutionary approach for gene selection and classification of microarray data based on SVM error-bound theories, *Biosystems*, 100(1), 39-46.

Dechter, R. and Rish, I. (1994) Directional resolution: The davis-putnam procedure, revisited, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, 134–145.

Delmont, TO., Robe, P., Cecillon, S., Clark, IM., Constancias, F., Simonet, P. et al. (2011) Accessing the soil metagenome for studies of microbial diversity, *Appl Environ Microbiol*, 77, 1315-1324.

DeLong, E., Wickham, G. and Pace, N. (1989) Phylogenetic stains: ribosomal RNA-based probes for the identification of single cells, *Science*, 243, 1360-1363.

Denonfoux, J., Parisot, N., Dugat-Bony, E., Biderre-Petit, C., Boucher, D., Morgavi, DP., Le Paslier, D., Peyretaillade, E. and Peyret, P. (2013) Gene capture coupled to high-throughput sequencing as a strategy for targeted metagenome exploration, *DNA Res.*, 20, 185–196.

DeSantis, TZ., Dubosarskiy, I., Murray, SR. and Andersen, GL. (2003) Comprehensive aligned sequence construction for automated design of effective probes (CASCADE-P) using 16S rDNA, *Bioinformatics*, 19, 1461-1468.

Desantis, TZ., Stone, CE., Murray, SR., Moberg, JP. and Andersen, GL. (2005) Rapid quantification and taxonomic classification of environmental DNA from both prokaryotic and eukaryotic origins using a microarray, *FEMS microbiology letters*, 245, 271-278.

DeSantis, TZ., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, EL., Keller, K. et al. (2006) Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB, *Appl Environ Microbiol*, 72, 5069-5072.

DeSantis, TZ., Brodie, EL., Moberg, JP., Zubieta, IX., Piceno, YM. and Andersen, GL. (2007) High-density universal 16S rRNA microarray analysis reveals broader diversity than typical clone library when sampling the environment, *Microb Ecol.*, 53, 371-383.

De Risi, JL., Iyer, VR. and Brown, PO. (1997) Exploring the metabolic and genetic control of gene expression on a genomic scale, *Science*, 278, 680-686.

Dharmadi, Y. and Gonzalez, R. (2004) DNA microarrays: experimental issues, data analysis, and application to bacterial systems, *Biotechnology progress*, 20, 1309-1324.

Diaw, S., Lbath, R. and Coulette, B. (2008) Etat de l'art sur le développement logiciel dirigé par les modèles, *Technique et Science Informatique TSI*, 29, 505-536.

Ding, Y., Chan, CY. and Lawrence, CE. (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble, *RNA*, 11, 1157–1166.

Dongarra, J. and Heroux, MA. (2013) Toward a New Metric for Ranking High Performance Computing Systems, *Technical Report SAND2013-4744, UTK EECS and Sandia National Labs http://www.netlib.org/utk/people/JackDongarra/PAPERS/HPCG-Benchmark-utk.pdf, accessed April 15, 2014*, 11p.

Dowd, SE., Zaragoza, J., Rodriguez, JR., Oliver, MJ. and Payton, PR. (2005) Windows .NET Network Distributed Basic Local Alignment Search Toolkit (W.ND-BLAST), *BMC Bioinformatics*, 6, 93.

Do, CB., Woods, DA. and Batzoglou, S. (2006) CONTRAfold: RNA secondary structure prediction without physics-based models, *Bioinformatics*, 22(14), e90–e98.

Do, JH. and Choi, DK. (2008) Clustering approaches to identifying gene expression patterns from DNA microarray data, *Mol Cells*, 25, 279–288.

Dudoit, S., Gentleman, RC. and Quackenbush, J. (2003) Open source software for the analysis of microarray data, *BioTechniques*, 34(3), Supplement: Microarrays in Cancer: Research and Applications, 45-51.

Dugat-Bony, E., Missaoui, M., Peyretaillade, E., Biderre-Petit, C., Bouzid, O., Gouinaud, C. et al. (2011) HiSpOD: probe design for functional DNA microarrays, *Bioinformatics*, 27, 641-648.

Dugat-Bony, E., Biderre-Petit, C., Jaziri, F., David, M., Denonfoux, J., Lyon, D. et al (2012a) In situ TCE degradation mediated by complex dehalorespiring communities during biostimulation processes, *Microbial biotechnology*, 5(5):642-53.

Dugat-Bony, E., Peyretaillade, E., Parisot, N., Biderre-Petit, C., Jaziri, F., Hill, D. et al. (2012b) Detecting unknown sequences with DNA microarrays: explorative probe design strategies, *Environmental microbiology*, 14, 356-371.

Dunteman, GH. (1989) Principal Components Analysis, *Sage Publications*, 96p.

D'Haeseleer, P., Liang, S. and Somogyi, R. (2000) Genetic Network inference: from co-expression clustering to reverse engineering, *Bioinformatics*, 16(8), 707-726.

Eddy, SR. and Durbin, R. (1994), RNA sequence analysis using covariance models, *Nucleic Acids Res.*, 22(11), 2079-2088.

Een, N. and Sörensson, N. (2004) An Extensible SAT-solver, Theory and Applications of Satisfiability Testing, *Lecture Notes in Computer Science*, 2919, 502-518.

Een, N. and Sörensson, N. (2005) MiniSat v1.13 - A SAT solver with conflict- clause minimization, *Poster In SAT 2005*.

Efron, B., Tibshirani, R., Storey, J. and Tusher, V. (2001) Empirical Bayes analysis of a microarray experiment, *J. Amer. Statist. Assoc.*, 96, 1151-1160.

Ehrenreich, A. (2006) DNA microarray technology for the microbiologist: an overview, *Appl Microbiol Biotechnol*, 73, 255-273.

Eisen, M., Spellman, P., Brown, P. and Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns, *Proc Natl Acad Sci USA*, 95, 14863-14868.

Evertsz, E.M., Au-Young, J., Ruvolo, M.V., Lim, A.C. and Reynolds, M.A. (2001) Hybridization crossreactivity within homologous gene families on glass cDNA microarrays, *Biotechniques*, 31, 1182, 1184, 1186 passim.

Favre, JM. (2004a) Towards a basic Theory to Model Driven Engineering, *3rd Workshop in Software Model Engineering, WiSME 2004*, 8p.

Favre, JM. (2004b) Foundations of Meta-Pyramids: Languages vs. Metamodels, Episode II: Story of Thotus the Baboon, *Dagstuhl Seminar 04101 on Language Engineering for Model-Driven Software Development (Dagsthul, Germany)*, 70-80.

Favre, JM., Estublier, J. and Blay-Fornarino, M. (2006) L'ingénierie dirigée par les modèles: au-delà du MDA, Hermès-Lavoisier (ed.), Paris, 236 p.

Feldhaar, H., Straka, J., Krischke, M., Berthold, K., Stoll, S., Mueller, MJ. and Gross, R. (2007) Nutritional upgrading for omnivorous carpenter ants by the endosymbiont Blochmannia, *BMC Biol*, 5, 48.

Feng, S. and Tillier, ERM. (2007) A fast and flexible approach to oligonucleotide probe design for genomes and gene families, *Bioinformatics*, 23, 1195-1202.

Ferguson, JW. and Towns, J. (2001) The Alliance Grid, *Advances in Engineering Software*, 32, 417-422.

Flanagan, J.L., Brodie, E.L., Weng, L., Lynch, S.V., Garcia, O., Brown, R. et al. (2007) Loss of bacterial diversity during antibiotic treatment of intubated patients colonized with Pseudomonas aeruginosa, *J. Clin Microbiol*, 45, 1954-1962.

Flynn, M.J. (1966) Very high speed computing systems, *Proceedings of the IEEE*, 54(12), 1901-1909.

Flynn, M. (1972) Some Computer Organizations and Their Effectiveness, *IEEE Transactions on Computers*, C-21(9), 948-960.

Foster, I. and Kesselman, C. (1997) Globus: A Metacomputing Infrastructure Toolkit, *International Journal on Supercomputer Applications*, 11, 115-128.

Foster, I. and Kesselman, C. (1999) The grid: blueprint for a new computing infrastructure, Morgan Kaufmann publishers, 677p.

Foster, I. (2001) The Globus Toolkit for Grid Computing, Proceedings of First IEEE/ACM International *Symposium on Cluster Computing and the Grid*, 2p.

Foster, I., Kesselman, C. and Tuecke, S. (2001) The Anatomy of the Grid - Enabling Scalable Virtual Organizations, *International Journal of High Performance Computing Applications*, 15, 200-222.

Foster, I. (2006) Globus Toolkit Version 4: Software for Service-Oriented Systems, *Journal of Computer Science and Technology*, 21, 513-520.

Franke-Whittle, IH., Goberna, M., Pfister, V. and Insam, H. (2009) Design and development of the ANAEROCHIP microarray for investigation of methanogenic communities, *J. Microbiol Methods*, 79, 279-288.

Fraune, S., Augustin, R., Anton-Erxleben, F., Wittlieb, J., Gelhaus, C., Klimovich, VB. et al. (2010) In an early branching metazoan, bacterial colonization of the embryo is controlled by maternal antimicrobial peptides, *Proc Natl Acad Sci USA*, 107, 18067-18072.

Fuhrman, S., Cunningham, MJ., Wen, X., Zweiger, G., Seilhamer, JJ. and Somogyi, R. (2000) The application of shannon entropy in the identification of putative drug targets, *Bio Systems*, 55, 5-14.

Fujita, A., Sato, JR., Rodrigues Lde, O., Ferreira, CE. and Sogayar, MC. (2006) Evaluating different methods of microarray data normalization, *BMC Bioinformatics*, 7, 469.

Furey, TS., Cristianini, N., Duffy, N., Bednarski, DW., Schummer, M. and Haussler, D. (2000), Support vector machine classification and validation of cancer tissue samples using microarray expression data, *Bioinformatics*, 16, 906-914.

Garrido, P., Gonzalez-Toril, E., Garcia-Moyano, A., Moreno-Paz, M., Amils, R. and Parro, V. (2008) An oligonucleotide prokaryotic acidophile microarray: its validation and its use to monitor seasonal variations in extreme acidic environments with total environmental RNA, *Environmental microbiology*, 10, 836-850.

Gentry, T.J., Wickham, G.S., Schadt, C.W., He, Z. and Zhou, J. (2006) Microarray applications in microbial ecology research, *Microb Ecol*, 52, 159-175.

George, D.G., Barker, W.C., and Hunt, L.T. (1986) The protein identification resource (PIR), *Nucleic Acids Res.*, 14, 11-15.

Ghebremedhin, B., Layer, F., Konig, W., Konig, B. (2008) Genetic Classification and Distinguishing of Staphylococcus Species Based on Different Partial gap, 16S rRNA, hsp60, rpoB, sodA, and tuf Gene Sequences, *Journal of Clinical Microbiology*, 46, 1019-1025.

Gîrdea, M., Noé, L. and Kucherov, G. (2010) Back-translation for discovering distant protein homologies in the presence of frameshift mutations, *Algorithms for Molecular Biology*, 5(1), 6.

Gittel, A., Sorensen, KB., Skovhus, TL., Ingvorsen, K., and Schramm, A. (2009) Prokaryotic community structure and sulfate reducer activity in water from high-temperature oil reservoirs with and without nitrate treatment, *Appl Environ Microbiol*, 75, 7086-7096.

Goldberg, E. and Novikov, Y. (2007) BerkMin: A fast and robust sat-solver, *Discrete Applied Mathematics*, 155 (12), 1549–1561.

Gorodkin, J., Heyer, LJ. and Stormo, GD. (1997), Finding the most significant common sequence and structure motifs in a set of RNA sequences, *Nucleic Acids Res.*, 25(18), 3724-3732.

Govindaraju, NK., Lloyd, B., Wang, W., Lin, MC. and Manocha, D. (2004) Fast Computation of Database Operations using Graphics Processors, *Proceedings of the ACM International Conference on Management of Data, Paris, France, 2004*, 215-226.

Gray, J., Tolvanen, JP., Kelly, S., Gokhale, A., Neema, S. and Sprinkle, J. (2007) Domain-Specific Modeling, *In CRC Handbook of Dynamic System Modeling, Fishwick, PA (ed.) CRC Press*, chapter 7, pp. 1-20.

Greenfield, J. and Short, K. (2003) Software Factories Assembling Applications with Patterns, Models, Frameworks and Tools, *Proceedings of 18th annual ACM SIGPLAN conference on Object-Oriented programming, systems, languages, and applications OOPSLA'03, Anaheim, Canada*, 16-27.

Grossi, F., Spizzo, R., Bordo, D., Cacitti, V., Valent, F., Rossetto, C. et al. (2010) Prognostic Stratification of Stage IIIA pN2 Non-small Cell Lung Cancer by Hierarchical Clustering Analysis of Tissue Microarray Immunostaining Data: An Alpe Adria Thoracic Oncology Multidisciplinary Group Study (ATOM 014), *J Thorac Oncol*., 5(9), 1354-60.

Gultyaev, AP. (1991) The computer simulation of RNA folding involving pseudoknot formation, *Nucleic Acids Res.*, 19, 2489-2494.

Gultyaev, AP., van Batenburg, FHD. and Pleij, CWA. (1995) The Computer Simulation of RNA Folding Pathways Using a Genetic Algorithm, *J. Mol. Biol.*, 250, 37-51.

Guschin, DY., Mobarry, BK., Proudnikov, D., Stahl, DA., Rittmann, BE. and Mirzabekov, AD. (1997) Oligonucleotide microchips as genosensors for determinative and environmental studies in microbiology, *Applied and environmental microbiology*, 63, 2397-2402.

Gutkin, M., Shamir, R. and Dror, G. (2009) SlimPLS: a method for feature selection in gene expression-based disease classification, *PLoS One*, 4(7), e6416.

Guyon, I., Weston, J., Barnhill, S. and Vapnik, V. (2002) Gene Selection for Cancer Classification using Support Vector Machines, *Machine Learning*, 46, 389-422.

Gu, J., Purdom, P., Franco, J., Wah, B. (1997) Algorithms for the Satisfiability problem: a survey, *DIMACS Series on Discrete Mathematics and Theoretical Computer Science, American Mathematical Society*, 35, 19-151.

Hahn, CK., Ross, KN., Warrington, IM., Mazitschek, R., Kanegai, CM., Wright, RD., Kung, AL., Golub, TR. and Stegmaier, K. (2008) Expression-based screening identifies the combination of histone deacetylase inhibitors and retinoids for neuroblastoma differentiation, *Proc Natl Acad Sci USA*, 105, 9751–9756.

Hammoudi, S. (2010) Contribution à l'étude et à l'application de l'ingénierie dirigée par les modèles, *Habilitation à diriger des recherches, Université d'Angers, France*, 114p.

Hanczar, B., Courtine, M., Benis, A., Henegar, C., Clément, K. and Zucker, JD. (2003), Improving classification of microarray data using prototype-based feature selection, *SIGKDD Explorations*, 5(2), 23-30.

Hanczar, B., Zucker, JD., Henegar, C. and Saitta, L. (2007) Feature construction from synergic pairs to improve microarray-based classification, *Bioinformatics*, 23(21), 2866-2872.

Hao, J., Lardeux, F., and Saubion, F. (2002) A hybrid genetic algorithm for the satisfiability problem, *Proceedings of the 1rst International Workshop on Heuristics, Beijing, China, 2002*, 102–109.

Harris, M., J., Coombe, G., Scheuermann, T. and Lastra, A. (2002) Physically-based visual simulation on graphics hardware, *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference On Graphics Hardware SaarbuCken, Germany, 2002*, 109-118.

Hartemink, A., Gifford, D., Jaakkola, T. and Young, R. (2001) Maximum likelihood estimation of optimal scaling factors for expression array normalization, *Proceedings of SPIE - The International Society for Optical Engineering*, 4266, 132-140.

Hartuv, E., Schmitt, A., Lange, J., Meier-Ewert, S., Lehrach, H. and Shamir, R. (1999) An algorithm for clustering cDNAs for gene expression analysis, *Proceedings of the third annual international conference on Computational molecular biology*, 188-197.

Heintzman, ND., Hon, GC., Hawkins, RD., Kheradpour, P., Stark, A., Harp, LF., Ye, Z., Lee, LK., Stuart, RK., Ching, CW., Ching, KA., Antosiewicz-Bourget, JE., Liu, H., Zhang, X., Green, RD., Lobanenkov, VV., Stewart, R., Thomson, JA., Crawford, GE., Kellis, M. and Ren, B. (2009) Histone modifications at human enhancers reflect global cell-type-specific gene expression, *Nature*, 459(7243),108-12.

Heitz, C., Thiemann, P. and Wölfle, T. (2007) Integration of an Action Language Via UML Action Semantics, *Trends in Enterprise Application Architecture, Lecture Notes in Computer Science*, 4473, 172-186.

Heljanko, K., Keinänen, M., Lange, M. and Niemelä, I. (2012) Solving parity games by a reduction to SAT, *Journal of Computer and System Sciences*, 78(2), 430–440.

Hemel, Z., Kats, LCL. and Visser, E. (2008) Code generation by model transformation: a case study in transformation modularity, *In Software and systems modeling, Springer Berlin/Heidelberg*, 9, 183-198.

Herrero, J., Valencia, A. and Dopazo, J. (2001) A hierarchical unsupervised growing neural network for clustering gene expression patterns, *Bioinformatics*, 17, 126-136.

He, Z., Gentry, T.J., Schadt, C.W., Wu, L., Liebich, J., Chong, S.C., Huang, Z., Wu, W., Gu, B., Jardine, P., Criddle, C. and Zhou, J. (2007) GeoChip: a comprehensive microarray for investigating biogeochemical, ecological and environmental processes, *ISME J.*, 1, 67-77.

He, Z., Deng, Y., Van Nostrand, J.D., Tu, Q., Xu, M., Hemme, C.L. et al. (2010) GeoChip 3.0 as a highthroughput tool for analyzing microbial community composition, structure and functional activity, *ISME J.*, 4, 1167-1179.

He, Z., Van Nostrand, J.D., Deng, Y. and Zhou, J.Z. (2011) Development and applications of functional gene microarrays in the analysis of the functional diversity, composition, and structure of microbial communities, *Front Environ Sci Engin China*, 5(1), 1-20.

Hill, DRC. (2006) Traduction inverse de Protéines, *LIMOS UMR CNRS 6158 Technical Report 2006, Novembre*, 8 p.

Hockney, RW. and Jesshope, CR. (1988) Parallel Computers 2: Architecture, Programming and Algorithms, *IOP Publ. Ltd., Bristol, UK*, 625p.

Hofacker, IL. (2003) Vienna RNA secondary structure server, *Nucleic Acid Res.*, 31(13), 3429-3431.

Hoffman, AR. and Traub, JF. (1989) Supercomputers: directions in technology and applications, *National Academies Press*, 102p.

Hoshida, Y., Villanueva, A., Kobayashi, M., Peix, J., Chiang, DY., Camargo, A. et al. (2008) Gene expression in fixed tissues and outcome in hepatocellular carcinoma, *N Engl J Med*, 359, 1995–2004.

Hovatta, I., Saharinen, J., Kimppa, K., Lehmussola, A., Pasanen, T., Saarela, J. et al. (2005) DNA microarray data analysis, *2nd edition, CSC - Scientific Computing Ltd., Finland*, 163p.

Howley, P.M., Israel, M.F., Law, M.-F. and Martin, M.A. (1979) A rapid method for detecting and mapping homology between heterologous DNAs. Evaluation of polyomavirus genomes, *Journal of Biolagical Chemistry*, 254, 4876-4883.

Hsu, AL., Tang, S. and Halgamuge, SK. (2003) An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data, *Bioinformatics*, 19(16), 2131–2140.

Huang, JC., Morris, QD., Hughes, TR. and Frey, BJ. (2005) GenXHC: a probabilistic generative model for cross-hybridization compensation in high-density genome-wide microarray data, *Bioinformatics*, 21, i222-i231.

Huang, HL. and Chang, FL. (2007) ESVM: Evolutionary support vector machine for automatic feature selection and classification of microarray data, *Biosystems*, 90(2), 516–528.

Hugenholtz, P. (2002) Exploring prokaryotic diversity in the genomic era, *Genome biology*, 3(2), reviews0003.1-0003.8.

Hughes, T.R., Mao, M., Jones, A.R., Burchard, J., Marton, M.J., Shannon, K.W. et al. (2001) Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer, *Nat Biotechnol*, 19, 342-347.

Huyghe, A., Francois, P., Charbonnier, Y., Tangomo-Bento, M., Bonetti, E.J., Paster, B.J., Bolivar, I., Baratti-Mayer, D., Pittet, D. and Schrenzel, J. (2008) Novel microarray design strategy to study complex bacterial communities, *Applied and environmental microbiology*, 74, 1876-1885.

Hwang, K. and Briggs, FA (1984) Computer architecture and parallel processing, *McGraw-Hill Inc., US*, 846p.

Iacob, ME, Steen, MWA. and Heerink, L. (2008) Reusable Model Transformation Patterns, *Proceedings of 12th Entreprise Distributed Object Computing Conference Workshops*, 1-10.

Iwama, K. (1989) Cnf satisfiability test by counting and polynomial average time, *SIAM J. Comput.*, 18, 385–391.

Jackson, D. and Vaziri, M. (2000) Finding bugs with a constraint solver, *Proceedings of the International Symposium on Software Testing and Analysis, Portland, OR, USA*, 14-25.

Jaziri, F., Missaoui, M., Cipière, S., Peyret, P. and Hill, D.R.C. (2011) Large Scale Parallelization Method of 16S rRNA Probe Design Algorithm on Distributed

Architecture: Application to Grid Computing, *Proceedings of the IEEE International Conference on Informatics and Computational Intelligence ICI2011, Bandung, Indonesia, 2011, ISBN: 978-0-7695-4618-6*, 35-40.

Jaziri, F., Parisot, N., Denonfoux, J., Dugat-Bony, E., Peyretaillade, E., Peyret, P. and Hill, DRC. (2012) MetaExploArrays : a large-scale oligonucleotide probe design software for explorative DNA microarrays, *Proceedings of the Thirteenth International Conference on Parallel and Distributed Computing, Applications and Technologies PDCAT2012, IEEE, Beijing, China, 2012, ISBN: 978-0-7695-4879-1*, 660-667.

Jaziri, F., Peyretaillade, E., Peyret, P. and Hill, DRC. (2013) Fast Distributed Computing for Complete Large Scale Backtranslation of Oligopeptides: Application to probe design, *Proceedings of the 2013 International Conference on High Performance Computing & Simulation, ACM/IEEE/IFIP, Helsinki July 1st-5th, 2013*, pp. 251-258.

Jaziri, F., Peyretaillade, E., Missaoui, M., Parisot, N., Cipière, S., Denonfoux, J., Mahul, A., Peyret, P. and Hill, DRC. (2014a) Large scale explorative oligonucleotide probe selection for thousands of genetic groups on a computing grid: application to phylogenetic probe design using a curated small subunit ribosomal RNA gene database, *Scientific World Journal, 2014*, doi: 10.1155/2014/350487, 9p.

Jaziri, F., Missaoui, M., Parisot N., Abid, A., Denonfoux, J., Ribiere, C., Boucher, D., Brugere, J.F., Mahul, A., Hill, DRC, Peyretaillade, E. and Peyret, P. (2014b) PhylODb: a 16S rRNA oligonucleotide probe database for prokaryotic identification, *Database: The Journal of Biological Databases and Curation*, 2014, 1-7, doi: 10.1093/database/bau036.

Jansson, JK. and Prosser, JI. (2013) Microbiology: The life beneath our feet, *Nature*, 494(7435), 40-1.

Johnson, DS. (1974) Fast algorithms for bin packing, *Journal of Computer and System Sciences*, 8(3), 272–314.

Kallio, MA., Tuimala, JT., Hupponen, T., Klemela, P., Gentile, M., Scheinin, AI., Koski, M., Korpelainen, EI., and Käki, J. (2011) Chipster: user-friendly analysis software for microarray and other high-throughput data, *BMC Genomics*, 12, 507.

Kanaris, I., Mylonakis, V., Chatziioannou, A., Maglogiannis, I. and Soldatos, J. (2009) HECTOR: Enabling Microarray Experiments over the Hellenic Grid Infrastructure, *Journal of Grid Computing*, 7(3), 395-416.

Kane, MD., Jatkoe, TA., Stumpf, CR., Lu, J., Thomas, JD. and Madore, SJ. (2000) Assessment of the sensitivity and specificity of oligonucleotide (50mer) microarrays, *Nucleic acids research*, 28, 4552-4557.

Kang, S., Denman, SE., Morrison, M., Yu, Z., Dore, J., Leclerc, M. and McSweeney, CS. (2010) Dysbiosis of fecal microbiota in Crohn's disease patients as revealed by a custom phylogenetic microarray, *Inflamm Bowel Dis.*, 16, 2034-2042.

Kang, J. and Xu, EY. (2013) An integrated hierarchical Bayesian approach to normalizing left-censored microRNA microarray data, *BMC Genomics*, 14, 507.

Katoh, K., Misawa, K., Kuma, K. and Miyata T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, *Nucleic Acids Res*, 30(14),3059-66.

Kautz, HA. and Selman, B. (1998) The role of domain-specific knowledge in the planning as satisfiability framework, *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)/Conference on Artificial Intelligence Planning Systems (AIPS)*, 181–189.

Kelly, JJ., Siripong, S., McCormack, J., Janus, LR., Urakawa, H., El Fantroussi, S., Noble, P.A., Sappelsa, L., Rittmann, B.E. and Stahl, D.A. (2005) DNA microarray detection of nitrifying bacterial 16S rRNA in wastewater treatment plant samples, *Water research*, 39, 3229-3238.

Kerr, M.K., Martin, M. and Churchill, G.A. (2000) Analysis of variance for gene expression microarray data, *Journal of Computational Biology*, 7, 819-837.

Kleppe, A., Warmer, S. and Bast, W. (2003) MDA Explained The Model Driven Architecture: Practice and Promise, *Addison-Wesley*, 192 p.

Klitgaard, K., Boye, M., Capion, N. and Jensen, TK. (2008) Evidence of multiple Treponema phylotypes involved in bovine digital dermatitis as shown by 16S rRNA gene analysis and fluorescence in situ hybridization, *J. Clin Microbiol*, 46, 3012-3020.

Knudsen, B. and Hein, J. (1999) RNA secondary structure prediction using stochastic context-free grammars and evolutionary history, *Bioinformatics*, 15, 446-454.

Knudsen, B. and Hein, J. (2003) Pfold: RNA secondary structure prediction using stochastic context-free grammars, *Nucleic Acids Res.*, 31, 3423-3428.

Kreil, D.P., Russell, R.R. and Russell, S. (2006) Microarray oligonucleotide probes, *Methods in enzymology*, 410, 73-98.

Kohonen, T. (1982) Self-organised formation of topologically correct feature map, *Biological Cybernetics*, 43, 56–69.

Kohonen, T. (2001) Self-Organizing Maps, *Springer Series in Information Sciences, Vol. 30, 3rd edition*, 501 p.

Koltai, H., and Weingarten-Baror, C. (2008) Specificity of DNA microarray hybridization: characterization, effectors and approaches for data correction, *Nucleic Acids Res*, 36, 2395-2405.

Konagaya, A. (2006) Trends in life science grid: from computing grid to knowledge grid, *BMC Bioinformatics*, 7, S10.

Konishi, F., Shiroto, Y., Umetsu, R. and Konagaya, A. (2003) Scalable BLAST Service in OBIGrid Environment, *Genome Informatics*, 14, 535-536.

Koschmieder, A., Zimmermann, K., Trissl, S., Stoltmann, T. and Leser, U. (2012) Tools for managing and analyzing microarray data, *Brief Bioinform*, 13(1), 46-60.

Krishnan, A. (2005) GridBLAST: a Globus-based high-throughput implementation of BLAST in a Grid computing framework, *Concurrency and computation: Practice and Experience*, 17, 1607–1623.

Kucho, Ki., Yoneda, H., Harada, M. and Ishiura, M. (2004) Determinants of sensitivity and specificity in spotted DNA microarrays with unmodified oligonucleotides, *Genes & Genetic Systems*, 79, 189-197.

Kuck DJ. (1978) The Structure of Computers and Computations, *John Wiley & Sons, Inc., New York, USA*, 634p.

Kumar, SM. and Joshi, RR. (2004) GBTK: a toolkit for grid implementation of BLAST, *Proceedings of 7th International Conference on High Performance Computing and Grid in Asia Pacific Region, Tokyo, Japan*, 378-382.

Kyselkova, M., Kopecky, J., Felfoldi, T., Cermak, L., Omelka, M., Grundmann, GL. et al. (2008) Development of a 16S rRNA gene-based prototype microarray for the detection of selected actinomycetes genera, *Antonie Van Leeuwenhoek*, 94(3), 439-453.

Lange, M. (2005) Solving parity games by a reduction to SAT, *Proceedings of International Workshop on Games in Design and Verification, GDV'05, 2005*, 14p.

Lano, K., Clark, D., 2008. Model Transformation Specification and Verification, *The 8th International Conference on Quality Software, Oxford, UK, 2008*, 45-54.

Lardeux, F., Saubion, and Hao, JK. (2006) Gasat: A genetic local search algorithm for the satisfiability problem, *Evolutionary Computation*, 14(2), 223–253.

Lassmann, T. and Sonnhammer, EL. (2005) Kalign--an accurate and fast multiple sequence alignment algorithm, *BMC Bioinformatics*, 6, 298.

Lee, N., Nielsen, P., Andreasen, K., Juretschko, S., Nielsen, J., Schleifer, K. et al. (1999). Combination of fluorescent in situ hybridization and microautoradiography-a new tool for structure-function analyses in microbial ecology, *Applied and environmental microbiology*, 65, 1289-1297.

Lee, Y. and Lee, CK. (2003) Classification of multiple cancer types by multicategory support vector machines using gene expression data, *Bioinformatics*, 19(9), 1132-9.

Lehner, A., Loy, A., Behr, T., Gaenge, H., Ludwig, W., Wagner, M. and Schleifer, K.H. (2005) Oligonucleotide microarray for identification of Enterococcus species, *FEMS microbiology letters*, 246, 133-142.

Leinonen, R., Akhtar, R., Birney, E., Bower, L., Cerdeno-Tárraga, A., Cheng, Y. et al. (2011) The European Nucleotide Archive, *Nucleic Acids Research*, 39, D28–31.

Lemkin, PF., Thornwall, GC., Walton, KD. and Hennighausen, L. (2000) The microarray explorer tool for data mining of cDNA microarrays: application for the mammary gland, *Nucleic Acids Res.*, 28, 4452-4459.

Lemoine, S., Combes, F. and Le Crom, S. (2009) An evaluation of custom microarray applications: the oligonucleotide design challenge, *Nucleic Acids Res.*, 37, 1726-1739.

Leparc, G.G., Tüchler, T., Striedner, G., Bayer, K., Sykacek, P., Hofacker, I.L. and Kreil, D.P. (2009) Model-based probe set optimization for high-performance microarrays, *Nucleic Acids Research*, 37, e18.

Letowski, J., Brousseau, R., and Masson, L. (2004) Designing better probes: effect of probe size, mismatch position and number on hybridization in DNA oligonucleotide microarrays, *J. Microbiol Methods*, 57, 269-278.

Leung, YF. and Cavalieri, D. (2003) Fundamentals of cDNA microarray data analysis, *Trends Genet.*, 19(11), 649-59.

Lewin, A., Johansen, J., Wentzel, A., Kotlar, H. K., Drabløs, F. and Valla, S. (2014) The microbial communities in two apparently physically separated deep subsurface oil reservoirs show extensive DNA sequence similarities, *Environmental Microbiology*, 16, 545–558.

Liles, MR., Turkmen, O., Manske, BF., Zhang, M., Rouillard, JM., George, I. et al. (2010) A phylogenetic microarray targeting 16S rRNA genes from the bacterial division Acidobacteria reveals a lineage-specific distribution in a soil clay fraction, *Soil Biol Biochem*, 42, 739-747.

Ling, C. and Benkrid, K. (2010) Design and implementation of a CUDA-compatible GPU-based core for gapped BLAST algorithm, *Procedia Computer Science*, 1(1), 495-504.

Lin, F. and Zhao, Y. (2004) ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers, *Artificial Intelligence*, 157(1–2), 115–137.

Lipshutz, RJ., Fodor, SP., Gingeras, TR. and Lockhart, DJ. (1999) High density synthetic oligonucleotide arrays, *Nat. Genet.*, 21, 20–24.

Liu, W., Schmidt, B. and Muller-Wittig, W. (2011) CUDA-BLASTP: Accelerating BLASTP on CUDA-Enabled Graphics Hardware, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6), 1678-1684.

Li, K.B., (2003) ClustalW-MPI: ClustalW analysis using distributed and parallel computing, *Bioinformatics*, 19, 1585-1586.

Li, W. and Godzik, A. (2006) Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, *Bioinformatics*, 22, 1658-1659.

Li, CM., Wei, W. and Zhang, H. (2007) Combining adaptive noise and look-ahead in local search for sat, *Theory and Applications of Satisfiability Testing - SAT 2007, Lecture Notes in Computer Science*, 4501, 121–133.

Li, T., Wu, T-D., Mazéas, L., Toffin, L., Guerquin-Kern, J-L., Leblon, G. et al. (2008) Simultaneous analysis of microbial identity and function using NanoSIMS, *Environmental microbiology*, 10, 580-588.

Li, C. and Fan, Y. (2013) CCA2013, *Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions*, 14-15.

Loy, A., Lehner, A., Lee, N., Adamczyk, J., Meier, H., Ernst, J., Schleifer, K.H. and Wagner, M. (2002) Oligonucleotide microarray for 16S rRNA gene-based detection of all recognized lineages of sulfate-reducing prokaryotes in the environment, *Applied and environmental microbiology*, 68, 5064-5081.

Loy, A., Schulz, C., Lucker, S., Schopfer-Wendels, A., Stoecker, K., Baranyi, C., Lehner, A. and Wagner, M. (2005) 16S rRNA gene-based oligonucleotide microarray for

environmental monitoring of the betaproteobacterial order “Rhodocyclales”, *Applied and environmental microbiology*, 71, 1373-1386.

Loy, A. and Bodrossy, L. (2006) Highly parallel microbial diagnostics using oligonucleotide microarrays, Clinica chimica acta, *international journal of clinical chemistry*, 363, 106-119.

Loy, A., Maixner, F., Wagner, M. and Horn, M. (2007) probeBase--an online resource for rRNA-targeted oligonucleotide probes: new features 2007, *Nucleic Acids Res.*, 35, D800–4.

Ludwig, W., Amann, R., Martinez-Romero, E., Schönhuber, W., Bauer, S., Neef, A. and Schleifer, KH. (1998) rRNA based identification and detection systems for rhizobia and other bacteria, *Plant Soil*, 204, 1-19.

Ludwig, W., Strunk, O., Westram, R., Richter, L., Meier, H., Kumar, Y. et al. (2004) ARB: a software environment for sequence data, *Nucleic Acids Res.*, 32, 1363-1371.

Lu, J., Getz, G., Miska, EA., Alvarez-Saavedra, E., Lamb, J., Peck, D., Sweet-Cordero, A., Ebert, BL., Mak, RH., Ferrando, AA., Downing, JR., Jacks, T., Horvitz, HR. and Golub, TR. (2005) MicroRNA expression profiles classify human cancers, *Nature*, 435, 834–838.

Lynce, I. and Marques-Silva, J. (2006) SAT in Bioinformatics: Making the Case with Haplotype Inference, *Proceedings of the Ninth International Conference on Theory and Applications of Satisfiability Testing (SAT’06), Lecture Notes in Computer Science*, 4121, 136-141.

Lyngso, RB., Zuker, M. and Pedersen, CNS. (1999) Fast evaluation of internal loops in RNA secondary structure prediction, *Bioinformatics*, 15, 440-445.

Mackay, A., Weigelt, B., Grigoriadis, A., Kreike, B., Natrajan, R., A'Hern, R., Tan, DS., Dowsett, M., Ashworth, A. and Reis-Filho, JS. (2011) Microarray-Based Class Discovery for Molecular Classification of Breast Cancer: Analysis of Interobserver Agreement, *J Natl Cancer Inst.*, 103(8), 662-73.

Maglogiannis, I., Chatzioannou, A., Soldatos, J., Mylonakis, V. and Kanaris, Y. (2007) An Application Platform Enabling High Performance Grid Processing of Microarray Experiments, *Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems, IEEE Computer Society*, 477-482.

Manavski, S. and Valle, G. (2008) CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment, *BMC Bioinformatics*, 9(Suppl. 2), S10.

Mann, HB. and Whitney, DR. (1947) On the test of whether one of two random variables is stochastically larger than the other, *Ann. Math. Statist.*, 18, 50-60.

Manolios, P., Galceran Oms, M. and Oliva Valls, S. (2007) Checking Pedigree Consistency with PCS, *Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science,* 4424,339-342.

Marcais, G. and Kingsford, C. (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers, *Bioinformatics*, 27,764-770.

Markham, NR., Zuker, M. (2008) UNAFold: software for nucleic acid folding and hybridization, *Methods Mol Biol.*, 453, 3-31.

Marmur, J. and Doty, P. (1962) Determination of the base composition of deoxyribonucleic acid from its thermal denaturation temperature, *Journal of Molecular Biology*, 5, 109-118.

Mathews, DH., Sabina, J., Zuker, M. and Turner, DH. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure, *J. Mol. Biol.*, 288, 911-940.

Mathews, DH. and Turner, DH. (2002), Dynalign: an algorithm for finding the secondary structure common to two RNA sequences, *J Mol Biol.*, 317(2), 191-203.

Mathews, DH., Disney, MD., Childs, JL., Schroeder, SJ., Zuker, M., Turner, DH. (2004) Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure, *Proceedings of the National Academy of Sciences of the USA*, 101(19), 7287-7292.

Mathews, DH. (2006) Revolutions in RNA Secondary Structure Prediction, *Journal of Molecular Biology*, 359, 526-532.

Mazure, B., Saïs, L. and Grégoire, E. (1997) Tabu search for sat, *Proceedings of the 14th American National Conference on Artificial Intelligence (AAAI)*, 281–285.

Ma, J., Zhou, T., Gu, W., Sun, X. and Lu, Z. (2002) Cluster analysis of the codon use frequency of MHC genes from different species, *Biosystems*, 65, 199-207.

McDonald, D., Price, MN., Goodrich, JK., Nawrocki, EP., DeSantis, TZ., Probst, A., Andersen, GL., Knight, R. and Hugenholtz, P. (2012) An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea, *ISME J.*, 6, 610–618.

McCaskill, JS. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure, *Biopolymers*, 29(6-7), 1105-1119.

Mehta, JP. and Rani, S. (2011) Software and tools for microarray data analysis, *Methods Mol Biol.*, 784, 41-53.

Mell, P. and Grance, T. (2011) The NIST Definition of Cloud Computing, *National Institute of Standards and Technology, U.S. Department of Commerce, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, accessed April 20, 2014*, 7p.

Mens, T. and Van Gorp, P. (2006) A Taxonomy of Model Transformation, *Electronic Notes in Theoretical Computer Science*, 152, 125-142.

Mernik, M., Heering, J. and Sloane, A. (2005) When and how to develop domain-specific Languages, *ACM Comput. Surv.*, 37(4), 316–344.

Metzker, ML. (2010) Sequencing technologies - the next generation, *Nature Reviews Genetics*, 11, 31-46.

Militon, C., Rimour, S., Missaoui, M., Biderre, C., Barra, V., Hill, DRC., Moné, A., Gagne, G., Meier, H., Peyretaillade, E. and Peyret, P. (2007) PhylArray: phylogenetic probe design algorithm for microarray, *Bioinformatics*, 23, 2550-2557.

Miller, LD., Smeds, J., George, J., Vega, VB., Vergara, L., Ploner, A., Pawitan, Y., Hall, P., Klaar, S., Liu, ET. and Bergh, J. (2005) An expression signature for p53 status in human breast cancer predicts mutation status, transcriptional effects, and patient survival, P*roc Natl Acad Sci USA*, 102(38), 13550-5.

Mirto, M., Fiore, S., Epicoco, I., Cafaro, M., Mocavero, S., Blasi, E. and Aloisio, G. (2008) A Bioinfomatics Grid Alignment Toolkit, *Future Generation Computer Systems*, 24, 752-762.

Missaoui, M., Militon, C., Hill, DRC., Gouinaud, C. and Peyret, P. (2006) PRT: Parallel program for a full backtranslation of oligopeptides, *LIMOS UMR CNRS 6158 Research Report 2006; June*, 19p.

Missaoui, M., Militon, C., Hill, DRC. and Peyret, P. (2007) Complete Backtranslation of Oligopeptides for Metabolic Pathways Exploration of Complex Environments Using Functional Microarrays, *Proceedings of the 7th IEEE International Conference On BioInformatics and BioEngineering, Boston, MA, 2007*, 1275-1279.

Missaoui, M., Hill, DRC. and Peyret, P. (2008) Comparison of Algorithms for a Complete Backtranslation of Oligopeptides, *International Journal of Computational Biology and Drug Design*, 1(1), 26-38.

Missaoui, M. (2009) Contributions algorithmiques à la conception de sondes pour biopuces à ADN en environnements parallèles, *Thèse de doctorat, Université Blaise Pascal Clermont Ferrand, France http://tel.archives-ouvertes.fr/docs/00/72/45/65/PDF/2009CLF21994.pdf, accessed April 20, 2014*, 239p.

Mitchell, D., Selman, B. and Levesque, H. (1992) Hard and easy distributions of SAT problems, *Proceedings of the tenth national conference on Artificial intelligence AAAI'92*, 459-465.

Moller, I., Marcus, SE., Haeger, A., Verhertbruggen, Y., Verhoef, R., Schols, H., Ulvskov, P., Mikkelsen, JD., Knox, JP. and Willats, W. (2008) High-throughput screening of monoclonal antibodies against plant cell wall glycans by hierarchical clustering of their carbohydrate microarray binding profiles, *Glycoconjugate Journal*, 25(1), 37-48.

Mora, C., Tittensor, DP., Adl, S., Simpson, AGB. and Worm, B. (2011) How Many Species Are There on Earth and in the Ocean?, *PLoS Biol*, 9(8), e1001127.

Morales, SE., Holben, WE. (2011) Linking bacterial identities and ecosystem processes: can 'omic' analyses be more than the sum of their parts?, *FEMS Microbiology Ecology*, 75(1), 2-16.

Moreira, A. (2004) Genetic algorithms for the imitation of genomic styles in protein backtranslation, *Theoretical Computer Science*, 322(2), 297-312.

Moreira, A. and Maass, A. (2004) TIP: protein backtranslation aided by genetic algorithms, *Bioinformatics*, 20(13), 2148-2149.

Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L. and Malik, S. (2001) Chaff: Engineering an Efficient SAT Solver, *Proceedings of the 38th annual Design Automation Conference DAC 2001, Las Vegas*, 530-535.

Muhling, M., Woolven-Allen, J., Murrell, JC. and Joint, I. (2008) Improved group-specific PCR primers for denaturing gradient gel electrophoresis analysis of the genetic diversity of complex microbial communities, *ISME J.*, 2, 379-392.

Mukherjee M., Tamago P., Mesirov J.P., Slorim D., Verni A., Poggio T. (1999) Support vector machine classification of microarray data, *Technical Report, Massachusetts Institute of Technology, Cambridge, MA, A.I. Memo No.1677, C.B.C.L No.182*, 10p.

Mukherjee S. (2003) Classifying microarray data using support vector machines, *In: A Practical Approach to Microarray Data Analysis, Berrar, DP., Dubitzky, W. and Granzow M. (editors), Boston, MA: Kluwer Academic Publishers*, 166-185.

Munroe, DJ. and Harris, TJR. (2010) Third-generation sequencing fireworks at Marco Island, *Nature Biotechnology*, 28, 426-428.

Nadon, R. and Shoemaker, J. (2002) Statistical issues with microarrays: processing and analysis, *Trends Genet.*, 18(5), 265-71.

Nanni, L. and Lumini, A. (2011) Wavelet selection for disease classification by DNA microarray data, *Expert Systems with Applications*, 38(1), 990–995.

Nash, JA. (1993) computer program to calculate and design oligonucleotide primers from amino acid sequences, *Computer applications in the biosciences*, 9, 469-471.

Neufeld, JD., Mohn, WW. and de Lorenzo, V. (2006) Composition of microbial communities in hexachlorocyclohexane (HCH) contaminated soils from Spain revealed with a habitat-specific microarray, *Environ Microbiol*, 8, 126-140.

Nichols, D., Cahoon, N., Trakhtenberg, EM., Pham, L., Mehta, A., Belanger, A. et al. (2010) Use of Ichip for High-Throughput In Situ Cultivation of "Uncultivable" Microbial Species, *Applied and Environmental Microbiology*, 76, 2445-2450.

Ninio, J. (1979) Prediction of pairing schemes in RNA molecules-loop contributions and energy of wobble and non-wobble pairs, *Biochimie*, 61(10), 1133-1150.

Nirenberg, M. (2004) Historical review: Deciphering the genetic code--a personal account, *Trends in biochemical sciences*, 29(1), 46-54.

Nussinov, R. and Jacobson, AB. (1980) Fast algorithm for predicting the secondary structure of single stranded RNA, *Proceedings of the National Academy of Sciences of the USA*, 77(11), 6309-6313.

Oksanen, M. and Pietarinen, J. (2004) Phylosophy and biodiversity, Cambridge *University Press*, 276p.

OMG « Object Management Group » (2002), Meta Object Facility (MOF) Specification, *http://www.omg.org/spec/MOF/1.4/, accessed December 10, 2013*, 358 p.

Oshlack, A., Emslie, D., Corcoran, LM. and Smyth, GK. (2007) Normalization of boutique two-color microarrays with a high proportion of differentially expressed probes, *Genome Biol*, 8(1), R2.

Pace, NR. (1997) A Molecular View of Microbial Diversity and the Biosphere. *Science*, 276, 734-740.

Paliy, O., Kenche, H., Abernathy, F., and Michail, S. (2009) High-throughput quantitative analysis of the human intestinal microbiota with a phylogenetic microarray, *Appl Environ Microbiol*, 75, 3572-3579.

Palmer, C., Bik, EM., Eisen, MB., Eckburg, PB., Sana, TR., Wolber, PK., Relman, DA. and Brown, PO. (2006) Rapid quantitative profiling of complex microbial populations, *Nucleic acids research*, 34, e5.

Panjkovich, A. and Melo, F. (2005) Comparison of different melting temperature calculation methods for short DNA sequences, *Bioinformatics*, 21, 711-722.

Papanicolaou, C., Gouy, M. and Ninio, J. (1984) An energy model that predicts the correct folding of both the tRNA and the 5s RNA molecules, *Nucleic Acids Res.*, 12, 31-44.

Pareek, CS., Smoczynski, R. and Tretyn, A. (2011) Sequencing technologies and genome sequencing, *Journal of Applied Genetics*, 52, 413-435.

Pariset, L., Chillemi, G., Bongiorni, S., Romano Spica, V. and Valentini, A. (2009) Microarrays and highthroughput transcriptomic analysis in species with incomplete availability of genomic sequences, *Nat Biotechnol*, 25, 272-279.

Parisot, N., Denonfoux, J., Dugat-Bony, E. et al. (2012) KASpOD–a web service for highly specific and explorative oligonucleotide design, *Bioinformatics*, 28, 3161–3162.

Park, T., Yi, SG., Kang, SH., Lee, S., Lee, YS. and Simon, R. (2002) Evaluation of normalization methods for microarray data, *BMC Bioinformatics*, 4, 33.

Parmigiani G, Garrett ES, Irizarry RA, Zeger SL (eds) (2003) The Analysis of Gene Expression Data: Methods and Software, *Springer, New York*, 456p, doi:10.1007/b97411.

Pavlidis, P., Weston, J., Cai, J. and Noble, WS. (2002) Learning gene functional classifications from multiple data types, *J Comput Biol.*, 9, 401-411.

Pearson, WR. and Lipman, DJ. (1988) Improved tools for biological sequence comparison, *Procedings of the National Academy of Science*, 85, 2444-2448.

Peplies, J., Lau, SCK., Pernthaler, J., Amann, R., and Glöckner, FO. (2004) Application and validation of DNA microarrays for the 16S rRNA-based analysis of marine bacterioplankton, *Environ Microbiol*, 6, 638-645.

Pesole, G., Attimonelli, M. and Liuni, S. (1988) A backtranslation method based on codon usage strategy, *Nucleic Acids Res.*, 16, 1715-1728.

Pipatsrisawat, K. and Darwiche, A. (2006) RSat 1.03: SAT solver description, *Technical report D-152, Automated Reasoning Group, Computer Science Department, UCLA*, 2p.

Porro, I., Torterolo, L., Corradi, L., Fato, M., Papadimitropoulos, A., Scaglione, S., Schenone, A. and Viti, F. (2007) A Grid-based solution for management and analysis of microarrays in distributed experiments, *BMC Bioinformatics*, 8, S7.

Potlapally, NR., Raghunathan, A., Ravi, S., Jha, NK. and Lee, RB. (2007) Aiding side-channel attacks on cryptographic software with satisfiability-based analysis, *IEEE Trans. On VLSI Syst.*, 15(4), 465–470.

Pozhitkov, AE., Noble, PA., Domazet-Loso, T., Nolte, AW., Sonnenberg, R., Staehler, P. et al. (2006) Tests of rRNA hybridization to microarrays suggest that hybridization characteristics of oligonucleotide probes for species discrimination cannot be predicted, *Nucleic Acids Res*, 34, e66.

Pozhitkov, AE., Tautz, D. and Noble, PA. (2007) Oligonucleotide microarrays: widely applied--poorly understood, *Briefings in functional genomics & proteomics*, 6, 141-148.

Prüfer, K., Stenzel, U., Dannemann, M., Green, RE., Lachmann, M. and Kelso, J. (2008) PatMaN: rapid alignment of short sequences to large databases, *Bioinformatics*, 24(13), 1530-1.

Quackenbush, J. (2002) Microarray data normalization and transformation, *Nature Genetics*, 32, 496-501.

Quast, C., Pruesse, E., Yilmaz, P., Gerken, J., Schweer, T., Yarza, P., Peplies, J., Glöckner, FO. (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools, *Nucleic Acids Res.*, 41, D590–D596.

Quince, C., Curtis, TP., Sloan, WT. (2008) The rational exploration of microbial diversity, *ISME J.*, 2, 997-1006.

Rahim, L., and Mansoor, S., (2008) Proposed Design Notation for Model Transformation, *Proceedings of 19th Australian Conference on Software Engineering, Perth, Australia, 2008*, 589-598.

Rajan, S., Pena, JR., Jegga, AG., Aronow, BJ., Wolska, BM. and Wieczorek, DF. (2013) Microarray analysis of active cardiac remodeling genes in a familial hypertrophic cardiomyopathy mouse model rescued by a phospholamban knockout, *Physiol Genomics.*, 45(17), 764-73.

Rajilic-Stojanovic, M., Heilig, HG., Molenaar, D., Kajander, K., Surakka, A., Smidt, H., and de Vos, WM. (2009) Development and application of the human intestinal tract chip, a phylogenetic microarray: analysis of universally conserved phylotypes in the abundant microbiota of young and elderly adults, *Environ Microbiol*, 11, 1736-1751.

Rastogi, G., Osman, S., Vaishampayan, P.A., Andersen, G.L., Stetler, L.D. and Sani, R.K. (2010a) Microbial diversity in uranium mining-impacted soils as revealed by highdensity 16S microarray and clone library, *Microb Ecol*, 59, 94-108.

Rastogi, G., Osman, S., Kukkadapu, R., Engelhard, M., Vaishampayan, PA., Andersen, GL., and Sani, RK. (2010b) Microbial and mineralogical characterizations of soils collected from the deep biosphere of the former Homestake gold mine, South Dakota., *Microb Ecol*, 60, 539-550.

Rastogi, G., Barua, S., Sani, R.K., and Peyton, B.M. (2011) Investigation of Microbial Populations in the Extremely Metal-Contaminated Coeur d'Alene River Sediments, *Microb Ecol*, 62, 1-13.

Raychaudhuri, S., Stuart, JM. and Altman, RB. (2000) Principal components analysis to summarize microarray experiments: application to sporulation time series, *Pac Symp Biocomput.*, 455-466.

Relogio, A., Schwager, C., Richter, A., Ansorge, W., and Valcarcel, J. (2002) Optimization of oligonucleotidebased DNA microarrays, *Nucleic Acids Res*, 30, e51.

Rhee, S.K., Liu, X., Wu, L., Chong, S.C., Wan, X. and Zhou, J. (2004) Detection of genes involved in biodegradation and biotransformation in microbial communities by using 50-mer oligonucleotide microarrays, *Applied and environmental microbiology*, 70, 4303-4317.

Rimour, S., Hill, D., Militon, C. and Peyret, P. (2005) GoArrays: highly dynamic and efficient microarray probe design, *Bioinformatics*, 21, 1094-1103.

Rivas, E. and Eddy, S. (1999) A dynamic programming algorithm for RNA structure prediction including pseudoknots, *J. Mol. Biol.*, 285, 2053-2068.

Rizk, G. and Lavenie, D. (2009) GPU Accelerated RNA Folding Algorithm, Proceedings of the 9th International Conference on Computational Science, *Lecture Notes in Computer Science*, 5544, 1004–1013.

Robinson, J. A. (1965) A machine-oriented logic based on the resolution principle, *JACM*, 12, 23–81.

Rocke, DM., Ideker, T., Troyanskaya, O., Quackenbush J. and Dopazo, J. (2009) Papers on normalization, variable selection, classification or clustering of microarray data, *Bioinformatics*, 25(10), 701-702.

Rosenfeld, N., Aharonov, R., Meiri, E., Rosenwald, S., Spector, Y., Zepeniuk, M. et al. (2008) MicroRNAs accurately identify cancer tissue origin, *Nat. Biotechnol.*, 26, 462–469.

Rose, TM., Schultz, ER., Henikoff, JG., Pietrokovski, S., McCallum, CM. and Henikoff, S. (1998) Consensus-degenerate hybrid oligonucleotide primers for amplification of distantly related sequences, *Nucleic Acids Res.*, 26, 1628-1635.

Rose, TM., Henikoff, JG. and Henikoff, S. (2003) CODEHOP (COnsensus-DEgenerate Hybrid Oligonucleotide Primer) PCR primer design, *Nucleic Acids Res.*, 31, 3763-3766.

Rouillard, JM., Zuker, M. and Gulari, E. (2003) OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach, *Nucleic Acids Res.*, 31, 3057-3062.

Rumpf, M. and Strzodka, R. (2001) Level set segmentation in graphics hardware, *Proceedings of IEEE International Conference on Image Processing, 2001*, 3, 1103-1006.

Rusch, A. and Amend, JP. (2004) Order-specific 16S rRNA-targeted oligonucleotide probes for (hyper) thermophilic archaea and bacteria, *Extremophiles*, 8, 357-366.

Saeed, AI., Bhagabati, NK., Braisted, JC., Liang, W., Sharov, V., Howe, EA., Li, J., Thiagarajan, M., White, JA. and Quackenbush, J. (2006) TM4 microarray software suite, *Methods Enzymol*, 411, 134-93.

Sagaram, U.S., DeAngelis, K.M., Trivedi, P., Andersen, G.L., Lu, S.E., and Wang, N. (2009) Bacterial diversity analysis of Huanglongbing pathogen-infected citrus, using PhyloChip arrays and 16S rRNA gene clone library sequencing, *Appl Environ Microbiol*, 75, 1566-1574.

Sais, L. (2008) Problème SAT: Progrès et Défis, *Hermes science publications, Paris, ISBN 2746218860*, 352p.

Saitou, N. and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.*, 4, 406-425.

Sait, M., Hugenholtz, P. and Janssen, P. (2002) Cultivation of globally distributed soil bacteria from phylogenetic lineages previously only detected in cultivation-independent surveys, *Environmental microbiology*, 4, 654-666.

Sanguin, H., Herrera, A., Oger-Desfeux, C., Dechesne, A., Simonet, P., Navarro, E., Vogel, T.M., Moenne-Loccoz, Y., Nesme, X. and Grundmann, G.L. (2006) Development and validation of a prototype 16S rRNA-based taxonomic microarray for Alphaproteobacteria, *Environmental microbiology*, 8, 289-307.

Sanguin, H., Kroneisen, L., Gazengel, K., Kyselková, M., Remenant, B., Prigent-Combaret, C., Grundmann, GL., Sarniguet, A. and Moënne-Loccoz, Y. (2008) Development of a 16S rRNA microarray approach for the monitoring of rhizosphere Pseudomonas populations associated with the decline of take-all disease of wheat, *Soil Biology and Biochemistry*, 40, 1028-1039.

Sanguin, H., Sarniguet, A., Gazengel, K., Moenne-Loccoz, Y. and Grundmann, GL. (2009) Rhizosphere bacterial communities associated with disease suppressiveness stages of take-all decline in wheat monoculture, *New Phytol*, 184, 694-707.

SantaLucia, J., Allawi, HT. and Seneviratne, PA. (1996) Improved nearest-neighbor parameters for predicting DNA duplex stability, *Biochemistry*, 35, 3555-3562.

Savage, RS., Heller, K., Xu, Y., Ghahramani, Z., Truman, WM., Grant, M., Denby, KJ. And Wild, DL. (2008) R/BHC: fast Bayesian hierarchical clustering for microarray data, BMC *Bioinformatics*, 10, 242, doi: 10.1186/1471-2105-10-242.

Saviozzi, S., Cordero, F., Lo Iacono, M., Novello, S., Scagliotti, GV. and Calogero, RA. (2006) Selection of suitable reference genes for accurate normalization of gene expression profile studies in non-small cell lung cancer, *BMC Cancer*, 6, 200.

Schaefer, T.J. (1978), The complexity of satisfiability problems, *Proceedings of the tenth annual ACM symposium on Theory of computing 1978*, 216-226.

Schatz, MC., Trapnell, C., Delcher, AL. and Varshney, A. (2007) High-throughput sequence alignment using Graphics Processing Units, *BMC Bioinformatics*, 8, 474.

Schatz, MC., Phillippy, AM., Gajer, P., DeSantis, TZ., Andersen, GL. and Ravel, J. (2010) Integrated Microbial Survey Analysis of Prokaryotic Communities for the PhyloChip Microarray, *Appl Environ Microbiol*, 76(16), 5636–5638.

Schena, M., Shalon, D., Davis, R. and Brown, P. (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray, *Science*, 270, 467-470.

Schliep, A. and Rahmann, S. (2006) Decoding non-unique oligonucleotide hybridization experiments of targets related by a phylogenetic tree, *Bioinformatics*, 22, e424-430.

Schönhuber, W., Fuchs, B., Juretschko, S. and Amann, R. (1997) Improved sensitivity of whole-cell hybridization by the combination of horseradish peroxidase-labeled oligonucleotides and tyramide signal amplification, *Applied and environmental microbiology*, 63, 3268-3273.

Schonmann, S., Loy, A., Wimmersberger, C., Sobek, J., Aquino, C., Vandamme, P. et al. (2009) 16S rRNA gene-based phylogenetic microarray for simultaneous identification of members of the genus Burkholderia, *Environ Microbiol*, 11, 779-800.

Schuchhardt, J., Beule, D., Malik, A., Wolski, E., Eickhoff, H., Lehrach, H. and Herzel, H. (2000) Normalization strategies for cDNA microarrays, *Nucleic Acids Res.*, 28(10), e47.

Sharma, S., Hsu, CH. and Feng, WC. (2006) Making a case for a Green500 list, *Proceedings of the 20th international conference on Parallel and distributed processing (*IPDPS'06), Rhodes, Greece, 25-29 April 2006, 8p.

Selman, B., Levesque, H. and Mitchell, D. (1992) A new method for solving hard satisfiability problems, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI)*, 440–446.

Selman, B., Kautz, HA. and Cohen, B. (1996) Local search strategies for satisfiability testing, *Cliques, Coloring and Satisfiability: the Second DIMACS Implementation Challenge*, 26, 521-532.

Simmler, H., Singpiel, H. and Männer, R. (2003) Real-time primer design for DNA chips, *Concurrency and computation: practice and experience*, 16, 855-872.

Skillicorn, D.B (1988) A Taxonomy for Computer Architectures, *Computer*, 21(11), 46-57.

Seidewitz, E. (2003) What models mean, *IEEE Software*, 20, 26-32.

Selic, B. (2003) The Pragmatics of Model-Driven Development, *IEEE Software*, 20(5), 19-25.

Selic, B. (2012) What will it take? A view on adoption of model-based methods in practice, *Software and systems Modeling*, 11(4), 513-526.

Sendall, S. and Kosaczynski, W. (2003) Model transformation: the heart and soul of model-driven software development, *IEEE software*, 20(5), 42-45.

Sessitsch, A., Hackl, E., Wenzl, P., Kilian, A., Kostic, T., Stralis-Pavese, N., Sandjong, B.T. and Bodrossy, L. (2006) Diagnostic microbial microarrays in soil ecology, *The New phytologist*, 171, 719-735.

Severgnini, M., Cremonesi, P., Consolandi, C., Caredda, G., De Bellis, G. and Castiglioni, B. (2009) ORMA: a tool for identification of species-specific variations in 16S rRNA gene and oligonucleotides design, *Nucleic Acids Res.*, 37, e109.

Shendure, J. and Ji, H. (2008) Next-generation DNA sequencing, *Nature Biotechnology*, 26, 1135-1145.

Shterev, ID., Jung, SH., George, SL. and Owzar, K. (2010) permGPU: Using graphics processing units in RNA microarray association studies, *BMC Bioinformatics*, 11, 329.

Slonim, DK. (2002) From patterns to pathways: gene expression data analysis comes of age, *Nat Genet.*, 32 Suppl, 502-508.

Smith, A., Veneris, AG. and Vigla, SA. (2004) Design diagnosis using Boolean satisfiability, *Proceedings of the 2004 Conference on Asia South Pacific Design Automation: Electronic Design and Solution Fair*, 218-223.

Sogin, ML., Morrison, HG., Huber, JA., Welch, DM., Huse, SM., Neal, PR. et al. (2006) Microbial diversity in the deep sea and the underexplored "rare biosphere", *Proceedings of the National Academy of Sciences*, 103, 12115-12120.

Sokal, RR. and Michener, CD. (1958) A statistical method for evaluating systematic relationships, *University of Kansas Scientific Bulletin*, 28, 1409-1438.

Spellman, PT., Sherlock, G., Zhang, MQ., Iyer, VR., Anders, K., Eisen, MB., Brown, PO., Botstein, D. and Futcher, B. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization, *Mol Biol Cell.*, 9, 3273-3297.

Steinberg, D., Budinsky, F., Paternostro, M. and Merks, E. (2008) EMF: Eclipse modeling framework 2nd edition, *Addison-Wesley Professional*, 744 p.

Sterling, T., Becker, DJ., Savarese, D., Dorband, JE., Ranawake, UA. and Packer, CV. (1995) BEOWULF: A Parallel Workstation For Scientific Computation, *Proceedings of the 24th International Conference on Parallel Processing, Illinois, USA*, 11-14.

Stothard, P. (2000) The sequence manipulation suite: JavaScript programs for analyzing and formatting protein and DNA sequences, *Biotechniques*, 28, 1102-1104.

Stralis-Pavese, N., Sessitsch, A., Weilharter, A., Reichenauer, T., Riesing, J., Csontos, J. et al. (2004) Optimization of diagnostic microarray for application in analysing landfill methanotroph communities under different plant covers, *Environ Microbiol*, 6, 347-363.

Stratowa, C. (2003) Novel Framework for Distributed Storage and Analysis of Microarray Data in the Terabyte Range: An Alternative to BioConductor, *Proceedings of the 3rd International Workshop on Distributed Statistical Computing 2003, Vienne, Austria*, 21p.

Stüber, K. (1986) Nucleic acid secondary structure prediction and display, *Nucleic Acids Res.*, 14, 317-326.

Sturn, A., Quackenbush, J. and Trajanoski, Z. (2002) Genesis: cluster analysis of microarray data, *Bioinformatics*, 18, 207–8.

Sun, Y., Zhao, S., Yu, H., Gao, G. and Luo, J. (2007) ABCGrid: Application for Bioinformatics Computing Grid, *Bioinformatics*, 23, 1175-1177.

Suzuki, MT. and Giovannoni, SJ. (1996) Bias caused by template annealing in the amplification of mixtures of 16S rRNA genes by PCR, *Appl Environ Microbiol*, 62, 625-630.

Sztipanovits, J. and Karsai, G. (1997) Model-Integrated Computing Environment, *ACM SIGSOFT Software Engineering Notes*, 22(5), 72-73.

Tanaka, TS., Jaradat, SA., Lim, MK., Kargul, GJ., Wang, X., Grahovac, MJ. et al. (2000) Genome-wide expression profiling of mid-gestation placenta and embryo using a 15,000 mouse developmental cDNA microarray, *Proc Natl. Acad. Sci. USA*, 97(16), 9127-32.

Terrat, S., Peyretaillade, E., Goncalves, O., Dugat-Bony, E., Gravelat, F., Mone, A. et al. (2010) Detecting variants with Metabolic Design, a new software tool to design probes for explorative functional DNA microarray development, *BMC Bioinformatics*, 11, 478.

Thompson, JD., Higgins, DG. and Gibson, TJ. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Res.*, 22, 4673-4680.

Tinoco, I., Uhlenbeck, OC. and Levine, MD. (1971) Estimation of secondary structure in ribonucleic acids, *Nature*, 230, 362-367.

Tothill, RW., Tinker, AV., George, J., Brown, R., Fox, SB., Lade, S., Johnson, DS., Trivett, MK., Etemadmoghadam, D., Locandro, B., Traficante, N., Fereday, S., Hung, JA., Chiew, YE. Haviv, I., Australian Ovarian Cancer Study Group, Gertig, D., DeFazio, A. and Bowtell, DD. (2008) Novel molecular subtypes of serous and endometrioid ovarian cancer linked to clinical outcome, *Clin. Cancer Res.*, 2008, 14(16), 5198-208.

Törönen, P., Kolehmainen, M., Wong, G. and Castren E. (1999) Analysis of gene expression data using self- organizing maps, *FEBS Lett.*, 451, 142-146.

Tottey, W., Denonfoux, J., Jaziri, F., Parisot, N., Missaoui, M., Hill, D., Borrel, G., Peyretaillade, E., Alric, M., Harris, HM., Jeffery, IB., Claesson, MJ., O'Toole, PW., Peyret, P. and Brugère, JF. (2013) "HuGChip", an explorative phylogenetic microarray for determining gut microbiome diversity at Family level, *PLoS One*, 8(5), e62544.

Treleaven, PC., Brownbridge, DR. and Hopkins, RP. (1982) Data-Driven and Demand-Driven Computer Architecture, *ACM Comput. Surv.*, 14(1), 93-143.

Trelles, O. (2000) On the parallelisation of bioinformatics applications, *Brief in Bioinformatics*, 2, 181-194.

Trombetti, GA., Merelli, I., Orro, A. and Milanesi, L. (2007) BGBlast: A BLAST Grid Implementation with Database Self-Updating and Adaptive Replication, *Studies in Health Technology and Informatics*, 126, 23-30.

Tsaregorodtsev, A., Garonne, V., Closier, J., Frank, M., Gaspar, C., van Herwijnen, E. et al. (2003) DIRAC-distributed infrastructure with remote agent control, *Proceedings of Conference for Computing in High Energy and Nuclear Physics, 24-28 March 2003, La Jolla, California*, TUAT006, 8p.

Tseng, GC., Oh, MK., Rohlin, L., Liao, JC. and Wong, WH. (2001) cDNA microarray analysis: quality filtering, channel normalization, models of variations and assessment of gene effects, *Nucleic Acids Res.*, 29, 2549-2557.

Tusher, VG., Tibshirani, R. and Chu, G. (2001) Significance analysis of microarrays applied to the ionizing radiation response, *Proc Natl Acad Sci USA*, 98(9), 5116-21.

Tu, Q., Yu, H., He, Z., Deng, Y., Wu, L., Van Nostrand, JD., Zhou, A., Voordeckers, J., Lee, YJ., Qin, Y., Hemme, CL., Shi, Z., Xue, K., Yuan, T., Wang, A. and Zhou, J. (2014) GeoChip 4: a functional gene-array-based high-throughput environmental technology for microbial community analysis, *Mol Ecol Resour*, doi: 10.1111/1755-0998.12239.

Valentini, G. (2002) Gene expression data analysis of human lymphoma using support vector machines and output coding ensembles, *Artificial Intelligence in Medicine*, 26(3), 281–304.

Valm, AM., Mark, WJL., Rieken, CW., Hasegawa, Y., Sogin, M., Oldenbourg, R. et al. (2011) Systems-level analysis of microbial community organization through combinatorial labeling and spectral imaging, *Proceedings of the National Academy of Sciences of the USA*, 108, 4152-4157.

Van Batenburg, FHD., Gultyaev, AP. and Pleij, CWA. (1995) An APL-programmed Genetic Algorithm for the Prediction of RNA Secondary Structure, *J. theor. Biol.*, 174, 269-280.

Van Deursen, A., Klint, P. and Visser, J. (2000) Domain-specific languages: an annotated bibliography, *ACM SIGPLAN Notices*, 35(6), 26–36.

Vapnik, V. (1998) Statistical Learning Theory, *John Wiley & Sons Inc (ed.), New York*, 732p.

Velev, MN. and Bryant, RE. (2001) Effective use of boolean satisfiability procedures in the formal verification of superscalar and VLIW, *Proceedings of the Design Automation Conference, New York, NY, USA, ACM Press*, 226–231.

Venter, JC. (2004) Environmental Genome Shotgun Sequencing of the Sargasso Sea, *Science*, 304, 66-74.

Völter, M. (2011) MD/DSL Best Practices, Update March 2011, *http://www.voelter.de/data/pub/DSLBestPractices-2011Update.pdf, accessed April 15, 2014*, 31p.

Vora, GJ., Meador, CE., Stenger, DA. and Andreadis, JD. (2004) Nucleic acid amplification strategies for DNA microarray-based pathogen detection, *Appl Environ Microbiol*, 70, 3047-3054.

Vouzis, PD. and Sahinidis, NV. (2011) GPU-BLAST: using graphics processors to accelerate protein sequence alignment, *Bioinformatics*, 27(2), 182-188.

Wagner, M., Smidt, H., Loy, A. and Zhou, J. (2007) Unravelling Microbial Communities with DNA-Microarrays: Challenges and Future Directions, *Microb Ecol*, 53(3), 498-506.

Wang, X. and Seed, B. (2003) Selection of oligonucleotide probes for protein coding sequences, *Bioinformatics*, 19, 796-802.

Wei, W. and Li, CM. (2009) Switching between two adaptive noise mechanisms in local search for sat, *Technical report, Solver Description, SAT competition 2009*, 3p.

Welch, BL. (1938) The significance of the difference between two means when the population variances are unequal, *Biometrika*, 29, 350-362.

Wernersson, R. and Pedersen, AG. (2003) RevTrans: multiple alignment of coding DNA from aligned amino acid sequences, *Nucleic Acids Res.*, 31, 3537-3539.

Wetmur, JG. (1991) DNA probes: applications of the principles of nucleic acid hybridization, *Critical reviews in biochemistry and molecular biology*, 26, 227-259.

Wheeler, T.J. and Kececioglu, J.D. (2007) Multiple alignment by aligning alignments, *Bioinformatics*, 23, i559-i568.

White, G. and Seffens, W. (1998) Using a neural network to backtranslate amino acid sequences, *Electronic Journal of Biotechnology*, 1, 17-18.

Whitman, WB., Coleman, DC. and Wiebe, WJ. (1998) Prokaryotes: the unseen majority, *Proc Natl Acad Sci USA*, 95(12), 6578-6583.

Wiese, KC. and Hendriks, A. (2006) Comparison of P-RnaPredict and mfold—algorithms for RNA secondary structure prediction, *Bioinformatics*, 22, 934-942.

Wiese, KC., Deschenes, AA. and Hendriks, AG. (2008) RnaPredict—An Evolutionary Algorithm for RNA Secondary Structure Prediction, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5, 25-41.

Wilcoxon, F. (1945) Individual comparisons by ranking methods, *Biometrics*, 1(6), 80-83.

Willey, J., Sherwood, L. and Woolverton, C. (2007) Prescott's Microbiology, *McGraw-Hill Science/Engineering/Math, 7 edition*, 1088p.

Wilson, EO. (ed.) (1988) Biodiversity, *National Academy Press, Washington, DC*, 521p.

Wilson, KH., Wilson, WJ., Radosevich, JL., DeSantis, TZ., Viswanathan, VS., Kuczmarski, TA. and Andersen, GL. (2002) High-density microarray of small-subunit ribosomal DNA probes, *Applied and environmental microbiology*, 68, 2535-2541.

Woese, CR. and Fox, GE. (1977) Phylogenetic structure of the prokaryotic domain: the primary kingdoms, *Proc Natl Acad Sci USA*, 74(11), 5088-90.

Wolfinger, RD., Gibson, G., Wolfinger, ED., Bennett, L., Hamadeh, H., Bushel, P., Afshari, C. and Paules, RS. (2001) Assessing gene significance from cDNA microarray expression data via mixed models, *Journal of Computational Biology*, 8(6), 625-637.

Woolf, PJ. and Wang, Y. (2000) A fuzzy logic approach to analyzing gene expression data, *Physiol Genomics*, 3, 9-15.

Woo, M., Neider, J., Davis, T. and Shreiner, D., (1999) OpenGL Programming Guide: the official guide to learning OpenGL, version 1.2. *Addison-Wesley Longman Publishing Co., Inc.*, 784p.

Wu, CH., Yeh, LSL., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y. and Hu, Z. (2003) The Protein Information Resource, *Nucl. Acids Res.*, 31(1), 345-347.

Wu, C., Carta, R. and Zhang, L. (2005) Sequence dependence of cross-hybridization on short oligo microarrays, *Nucleic Acids Research*, 33, e84.

Wu, Z. and Aryee, M.J. (2010) Subset quantile normalization using negative control features, *J. Comput. Biol.*, 17(10), 1385-95.

Xia, T, SantaLucia, J., Burkard, ME., Kierzek, R., Schroeder, SJ., Jiao, X., Cox, C. and Turner, DH. (1998) Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs, *Biochemistry*, 37(42), 14719-35.

Xing, EP. and Karp, RM. (2001) CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts, *Bioinformatics*, 17(Suppl 1), S306-15.

Xiong, H., Zhang, D., Martyniuk, CJ., Trudeau, VL. and Xia, X. (2008) Using generalized procrustes analysis (GPA) for normalization of cDNA microarray data, *BMC Bioinformatics*, 9, 25.

Yang, YH., Dudoit, S., Luu, P., Lin, DM., Peng, V., Ngai, J. and Speed, TP. (2002) Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation, *Nucleic Acids Res.*, 30, e15.

Yergeau, E., Schoondermark-Stolk, S.A., Brodie, EL., Dejean, S., DeSantis, TZ., Goncalves, O. et al. (2009) Environmental microarray analyses of Antarctic soil microbial communities, *ISME J.*, 3(3), 340-351.

Yeung, KY. and Ruzzo, W.L. (2001) Principal component analysis for clustering gene expression data, *Bioinformatics*, 17(9), 763-774.

Yin, H., Cao, L., Qiu, G., Wang, D., Kellogg, L., Zhou, J., Dai, Z. and Liu, X. (2007) Development and evaluation of 50-mer oligonucleotide arrays for detecting microbial populations in Acid Mine Drainages and bioleaching systems, *Journal of microbiological methods*, 70, 165-178.

Yu, Y., Lee, C., Kim, J., and Hwang, S. (2005) Group-specific primer and probe sets to detect methanogenic communities using quantitative real-time polymerase chain reaction, *Biotechnol Bioeng*, 89, 670-679.

Zengler, K. (2002) Cultivating the uncultured, *Proceedings of the National Academy of Sciences*, 99, 15681-15686.

Zengler, K., Walcher, M., Clark, G., Haller, I., Toledo, G., Holland, T. et al. (2005). High-throughput cultivation of microorganisms using microcapsules, *Methods in enzymology*, 397, 124-130.

Zhang, L., Madigan, CF., Moskewicz, MW. and Malik, S. (2001) Efficient conflict driven learning in boolean satisfiability solver, *Proceedings of the International Conference on Computer-Aided Design (ICCAD'01)*, 279–285.

Zhou, J. (2003) Microarrays for bacterial detection and microbial community analysis, *Current opinion in microbiology*, 6, 288-294.

Zhou, Z., Dou, ZX., Zhang, C., Yu, HQ., Liu, YJ., Zhang, CZ. and Cao, YJ. (2007) A strategy to optimize the oligo-probes for microarray-based detection of viruses, *Virol Sin*, 22, 326-335-335.

Zhu, D., Fofanov, Y., Willson, RC. and Fox, GE. (2006) A parallel computing algorithm for 16S rRNA probe design, *Journal of Parallel and Distributed Computing*, 66, 1546-1551.

Zomaya, AY. (2006) Parallel Computing for Bioinformatics and Computational Biology, *Wiley Series on Parallel and Distributed Computing*, 816p.

Zuker, M. and Stiegler, P. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, *Nucleic Acid Res.*, 9(1), 133-148.

Zuker, M. (1989) On finding all suboptimal foldings of an RNA molecule, *Science*, 244, 48-52.

Zuker, M. (2000) Calculating nucleic acid secondary structure, *Current Opinion in Structural Biology*, 10, 303-310.

Zuker, M. (2003) Mfold web server for nucleic acid folding and hybridization prediction, *Nucleic Acids Res*, 31, 3406-3415.

Liste des publications

Revues internationales (7):

[1] Jaziri, F., Parisot N., Abid, A., Denonfoux, J., Ribiere, C., Boucher, D., Brugere, J.F., Mahul, A., Hill, DRC, Peyretaillade, E. and Peyret, P. (2014b) PhylODb: a 16S rRNA oligonucleotide probe database for prokaryotic identification, *Database: The Journal of Biological Databases and Curation*, 2014, 1-7, doi: 10.1093/database/bau036.

[2] Jaziri, F., Peyretaillade, E., Missaoui, M., Parisot, N., Cipière, S., Denonfoux, J., Mahul, A., Peyret, P. and Hill, DRC. (2014a) Large scale explorative oligonucleotide probe selection for thousands of genetic groups on a computing grid: application to phylogenetic probe design using a curated small subunit ribosomal RNA gene database, *Scientific World Journal, 2014*, doi: 10.1155/2014/350487, 9p.

[3] Jaziri, F., Peyretaillade, E., Peyret, P. and Hill, DRC.(2014), High Performance Computing of Oligopeptides Complete Backtranslation applied to DNA microarray probe design, *Concurrency and Computation: Practice and Experience*, in press.

[4] Tottey, W., Denonfoux, J., **Jaziri, F.**, Parisot, N., Missaoui, M., Hill, D., Borrel, G., Peyretaillade, E., Alric, M., Harris, HM., Jeffery, IB., Claesson, MJ., O'Toole, PW., Peyret, P. and Brugère, JF. (2013) "HuGChip", an explorative phylogenetic microarray for determining gut microbiome diversity at Family level, *PLoS One*, 8(5), e62544.

[5] Dugat-Bony, E., Peyretaillade, E., Parisot, N., Biderre-Petit, C., **Jaziri, F.**, Hill, D. et al. (2012b) Detecting unknown sequences with DNA microarrays: explorative probe design strategies, *Environmental microbiology*, 14, 356-371.

[6] Dugat-Bony, E., Biderre-Petit, C., **Jaziri, F.**, David, M., Denonfoux, J., Lyon, D. et al (2012a) In situ TCE degradation mediated by complex dehalorespiring communities during biostimulation processes, *Microbial biotechnology*, 5(5):642-53.

[7] Boucher, D., Laffaire, JB., **Jaziri, F.**, David, C., Biderre-Petit, C., Duquenne, P., Peyretaillade, E. and Peyret, P. (2011), Bacterial community composition of biological degreasing systems and health risk assessment for workers, *Microbial ecology.*, 62(4), 868-81.

Conférences internationales IEEE/ACM avec comité de lecture et publication d'articles (3):

[8] Jaziri, F., Peyretaillade, E., Peyret, P. and Hill, DRC. (2013) Fast Distributed Computing for Complete Large Scale Backtranslation of Oligopeptides: Application to probe design, *Proceedings of the 2013 International Conference on High Performance Computing & Simulation, ACM/IEEE/IFIP, Helsinki July 1st-5th, 2013*, 251-258.

[9] Jaziri, F., Parisot, N., Denonfoux, J., Dugat-Bony, E., Peyretaillade, E., Peyret, P. and Hill, DRC. (2012) MetaExploArrays : a large-scale oligonucleotide probe design software for explorative DNA microarrays, *Proceedings of the IEEE Thirteenth International Conference on Parallel and Distributed Computing, Applications and Technologies PDCAT2012, IEEE, Beijing, China, 2012, ISBN: 978-0-7695-4879-1*, 660-667.

[10] Jaziri, F., Missaoui, M., Cipière, S., Peyret, P. and Hill, D.R.C. (2011) Large Scale Parallelization Method of 16S rRNA Probe Design Algorithm on Distributed Architecture: Application to Grid Computing, *Proceedings of the IEEE International Conference on Informatics and Computational Intelligence ICI2011, Bandung, Indonesia, 2011, ISBN: 978-0-7695-4618-6*, 35-40.