Contributions algorithmiques à la conception de sondes pour biopuces à ADN en environnements parallèles

Mohieddine Missaoui

N° d'ordre : D.U 1994
EDSPIC : 465

Université Blaise Pascal – CLERMONT II

ECOLE DOCTORALE DES SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

Mohieddine MISSAOUI

Pour obtenir le grade de

Docteur d'Université

SPECIALITE : Informatique

Contributions algorithmiques à la conception de sondes pour biopuces à ADN en environnements parallèles

Date de soutenance : 9 Décembre 2009

Membres du jury

Rapporteurs:	**Mr. Yvan Moënne-Loccoz**	**Professeur à l'Université de Lyon 1**
	Mr. Jean-Daniel Zucker	**Directeur de recherche à l'IRD**
Examinateurs:	**Mr. Vincent Breton**	**Professeur à l'Université Blaise Pascal**
	Mme. Cécile Militon	**Maître de conférences à l'Université de la Méditerranée- Aix-Marseille II**
Directeurs de thèse:	**Mr. David Hill**	**Professeur à l'Université Blaise Pascal**
	Mr. Pierre Peyret	**Professeur à l'Université Blaise Pascal**

Laboratoires d'accueil :
Laboratoire d'Informatique et de Modélisation et Optimisation des Systèmes (LIMOS)
Laboratoire Micro-organismes : Génomes et Environnement (LMGE)
Campus Universitaire des Cézeaux, 63170 Aubière, France

N° d'ordre : D.U 1994
EDSPIC : 465

Université Blaise Pascal – CLERMONT II

ECOLE DOCTORALE DES SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

Mohieddine MISSAOUI

Pour obtenir le grade de

Docteur d'Université

SPECIALITE : Informatique

Contributions algorithmiques à la conception de sondes pour biopuces à ADN en environnements parallèles

Date de soutenance : 9 Décembre 2009

Membres du jury

Rapporteurs:	**Mr. Yvan Moënne-Loccoz**	**Professeur à l'Université de Lyon 1**
	Mr. Jean-Daniel Zucker	**Directeur de recherche à l'IRD**
Examinateurs:	**Mr. Vincent Breton**	**Professeur à l'Université Blaise Pascal**
	Mme. Cécile Militon	**Maître de conférences à l'Université de la Méditerranée- Aix-Marseille II**
Directeurs de thèse:	**Mr. David Hill**	**Professeur à l'Université Blaise Pascal**
	Mr. Pierre Peyret	**Professeur à l'Université Blaise Pascal**

Laboratoires d'accueil :
Laboratoire d'Informatique et de Modélisation et Optimisation des Systèmes (LIMOS)
Laboratoire Micro-organismes : Génomes et Environnement (LMGE)
Campus Universitaire des Cézeaux, 63170 Aubière, France

Remerciements

Je souhaite remercier d'abord Mr Alain Quillot et Mr Christian Amblard pour m'avoir accueilli au sein des laboratoires LIMOS (Laboratoire d'Informatique et de Modélisation et Optimisation des Système – UMR CNRS 6158) et LMGE (Laboratoire Microorganismes : Génome et Envirennement – UMR CNRS 6023) et permis de travailler dans des bonnes conditions.

Je remercie mes deux co-directeurs de thèse, Pr. David Hill et Pr. Pierre Peyret, pour avoir encadré mes travaux de recherche durant cette thèse jusqu'à leur aboutissement. Je remercie David pour son dynamisme contagieux, pour son savoir, pour ses conseils précieux de tout ordre, pour son soutien et pour m'avoir donné une grande liberté dans la conduite de ces travaux. J'aimerais le féliciter pour son côté humain qui me touche depuis plusieurs années maintenant et qui ne fait qu'augenter mon estime pour lui. Merci de m'avoir aidé et poussé au plus haut pour me permettre de valoriser mes contributions dans des conférences internationales. Merci de m'avoir soutenu dans les moments difficiles. Je souhaite rendre hommage à Pierre pour m'avoir fait découvrir – et à mon esprit d'informaticien – le monde vaste de la biologie microbienne qui me passionnait auparavant. Je le remercie pour son professionnalisme, son ouverture d'esprit et pour sa rigueur à tous les niveaux. Je le remercie également pour m'avoir fait confiance et pour m'avoir toujours poussé à donner le meilleur de moi même. J'ai beaucoup appris grâce à vous deux.

Je tiens à remercier également tous les membres de l'équipe G2IM du laboratoire LMGE sans exception de m'avoir accueilli chaleureusement parmi eux comme le membre d'une famille. Mes remerciements vont ensuite au Pr. Vincent Breton pour la confiance qu'il m'a accordée et pour son professionalisme. Je remercie Mr. Yvan Moenne-Loccoz et Mr. Jean-Daniel Zucker d'avoir accepté de rapporter cette thèse.

Je remercie aussi mon ex-professeur d'administration des systèmes Christophe Gouinaud pour m'avoir supporté durant toutes ses années et pour ses connaissances précieuses en système. Et je remercie Pascale Gouinaud également pour tout son travail d'administration des architectures matérielles et logicielles utilisées durant cette thèse.

Je remercie toutes les personnes qui ont contribué de près ou de loin au bon déroulement de cette thèse. En particulier Brigitte Chebance, Béatrice Bourdieu, Eric Peyretaillade, Corinne Petit, Anne Moné, Cécile Militon, Matthieu Reichstadt, Jean Salzeman, Romain Reuillon et Jean-François Brugère.

Mes remerciements à tous ceux qui m'ont offert leur amitié au-delà de la thèse: Luc, Jonathan, Eric D, Abdel, Aurélie, Olivier, Séréna, Ourdia et Sébastien.

Enfin, je dédie cette thèse à mes parents en particulier et à ma famille en général ainsi qu'à mes deux tantes. A mon père, à qui je dois tout ce que je suis et à ma chère mère qui a été toujours là pour moi. A tous mes frères et sœurs Besma, Kamel, Asma et Ali. Une pensée profonde à tous mes amis.

Résumé

Les microorganismes constituent la plus grande diversité du monde vivant et restent encore largement méconnus. La compréhension du fonctionnement des écosystèmes et les rôles joués par les microorganismes reste un enjeu majeur de l'écologie microbienne. Le développement de nouvelles approches de génomique permet d'appréhender la complexité de ces systèmes. Dans ce contexte, les puces à ADN représentent des outils à haut débit de choix capables de suivre l'expression ou la présence de plusieurs milliers de gènes en une seule expérience. Cependant, l'une des étapes les plus difficiles, de part sa complexité et la quantité de données à traiter, est la sélection de sondes qui doivent être à la fois sensibles et spécifiques. Pour répondre à ces besoins en termes de performance et de qualité, d'une part, nous avons développé un algorithme de conceptions de sondes pour biopuces phylogénétiques que nous avons entièrement déployé sur la grille de calcul européenne EGEE, et d'autre part, nous avons proposé un deuxième algorithme pour biopuces fonctionnelles que nous avons déployé sur un cluster. Les deux algorithmes ont nécessité une phase importante d'ingénierie logicielle. Nous avons donc proposé une démarche d'ingénierie dirigée par les modèles (IDM) et notamment de transformation de modèle pour résoudre le problème de traduction inverse d'oligopeptide. Les deux algorithmes sont destinés à une utilisation massive et permettent de concevoir tout type de sondes.

Mots-clés: Conception de sondes, Puces à ADN, Grille de calcul, IDM, Ecologie microbienne.

Abstract

Microorganisms represent the largest diversity of the living beings and still largely unknown. Understanding ecosystems working mechanisms and the roles of microorganisms is a great challenge in microbial ecology. The development of new genomic approaches allows us to comprehend the complexity of those systems. In this context, DNA microarrays represent high-throughput tools able to show the expression or the presence of several thousands of genes on a single experiment. However, one of the hardest steps in microarray utilization is the probe selection because of its complexity and the huge amount of data to compute. Obtained probes should be specific and sensitive. To reach the goals of performance and quality, we have developed an algorithm for phylogenetic microarrays and we have deployed it on the European Computing Grid EGEE. We have also developed a new algorithm for functional microarrays and we have deployed it onto a cluster. The two algorithms needed an important step of software engineering. In this context, we have proposed a new method based on model driven engineering (MDE) and particularly on model transformation to solve the problem of backtranslation of oligopeptides. The two algorithms are intended to be used massively and allow bioinformaticians and biologists to design all kinds of probes.

Keywords: Probe design, Microarrays, Grid computing, MDE, Microbial ecology.

Table des matières

Tables des figures

Tables des tableaux

Introduction Générale

La pollution de notre planète est due essentiellement à certaines activités humaines engendrant des produits chimiques qui sont déversés accidentellement ou non dans la nature. L'ARIA (Analyse, Recherche et Information sur les Accidents) répertorie plus de 30 000 sites touchés par la pollution suite à des accidents dont 40% auraient entrainé la contamination de sites terrestres ou aquatiques. Le BARPI (Bureau d'Analyse des Risques et Pollutions Industrielles) est chargé de rassembler et de diffuser des données sur le retour d'expérience en matière d'accidents technologiques. Le ministère des affaires étrangères et européennes dresse un bilan des sols pollués en France depuis 2005 et emploie une politique de prévention, de traitement et de réhabilitation de ces sols et d'information de la population.

Cette pollution est souvent traitée par des procédés chimiques ou mécaniques pouvant être très invasifs et dangereux pour l'environnement. Mais une des découvertes de ces dernières décennies est le potentiel microbien dans la bioremédiation (décontamination par des microorganismes). En effet, certains sites pollués sont traités à l'aide de techniques impliquant des procédés faisant intervenir des microorganismes. Afin de préserver les écosystèmes et mieux exploiter les capacités de ces microorganismes à dégrader ou fixer ces polluants (hydrocarbures, solvants chlorés, métaux lourds, …), il est important de connaître la biodiversité microbienne et son évolution au cours du temps en présence de ces xénobiotiques. La difficulté réside bien sûr dans l'extrême diversité des microorganismes et le fait qu'une partie infime de ceux-ci est connue à ce jour. En effet, la quantité de microorganismes pouvant exister dans un site donné est inestimable mais certains auteurs, comme Wackett et Herchberger (2000), affirment qu'il existe environ 10 milliards de bactéries par gramme de terre, représentant 10 000 espèces différentes. D'autres estimations ont été réalisées mais, aujourd'hui, il n'est pas encore possible de conclure sur l'ampleur de la diversité microbienne dans les écosystèmes du fait de l'extrême diversité du monde microbien et de la difficulté à définir la notion « d'espèce » chez les microorganismes (Vieites et al., 2008).

Devant cette diversité extrême, l'identification des microorganismes et de leurs capacités métaboliques (lien structure-fonction) reste un enjeu majeur de l'écologie microbienne. Les gènes codant pour les ARN ribosomiques sont généralement choisis comme biomarqueurs dans les techniques d'identification car ils sont ubiquistes, ne subissent peu ou pas de transfert latéral, peuvent être isolés facilement grâce à la présence de régions conservées, présentent des régions variables assurant la discrimination des microorganismes et sont répertoriés dans des bases de données internationales permettant des recherches de similarité efficaces. Ces caractéristiques en font un bon marqueur moléculaire taxonomique.

Plus récemment, des techniques dites de méta-génomique et méta-transcriptomique ont été développées pour permettre de découvrir de nouveaux microorganismes et de nouvelles fonctions métaboliques en exploitant l'information génétique globale (Vieites et al., 2008). Ces approches utilisant souvent les techniques de séquençage ultra-haut débit comme le pyroséquençage restent onéreuses pour l'exploration d'environnements complexes et ne permettent pas de reconstruire l'image complète d'un écosystème (difficulté d'inventorier la totalité de la diversité, difficulté de traitements des masses de données générées). D'autres avancées significatives ont été réalisées avec le développement des biopuces ADN permettant de contribuer à une meilleure connaissance du fonctionnement des écosystèmes.

Apparues en milieu des années 90 (Schena et al., 1995), les biopuces ADN ont été mises au point pour l'étude de l'expression de la totalité des gènes d'un organisme. Cette technique connaît des développements récents en écologie microbienne (DeSantis et al., 2007 ; Moisander et al., 2006 ; Peplies et al., 2006 ; Sanguin et al., 2006). En effet, elle permet de détecter simultanément des milliers d'espèces bactériennes d'environnements complexes et/ou d'étudier la dynamique des communautés bactériennes et de mieux comprendre les mécanismes d'adaptation de ces microorganismes.

L'une des étapes clef du développement d'une biopuce à ADN est la détermination des sondes. En effet, pour considérer une sonde, elle doit vérifier des critères bien précis pour qu'elle puisse détecter spécifiquement sa cible et ce avec la meilleure sensibilité possible. Plusieurs logiciels de conception de sondes ADN ont été développés ces dernières années pour répondre aux exigences des biopuces ADN (Lemoine et al., 2009). Cependant, la conception de sondes pour biopuces ADN en écologie microbienne doit répondre à des spécificités particulières du fait de la complexité des environnements étudiés et doit faire face à la croissance exponentielle du nombre de séquences déposées dans les bases de données internationales.

Ainsi, les temps de calcul des logiciels (CommOligo (Li et al., 2005) ; ROSO (Reymond et al., 2004) ou PhylArray (Militon et al., 2007)) nécessitent plusieurs heures pour assurer la sélection des sondes pour un seul gène. De ce fait, en plus de leur complexité, les algorithmes de sélection de sondes doivent assurer de hauts niveaux de performances sous peine d'être inutilisables. Pour répondre aux besoins des applications dans différents domaines en termes de temps de calcul et d'espace de stockage, une des solutions apportées par l'informatique ces dernières années est le développement d'architectures de calcul à haute performance (High Performance Computing) qui deviennent financièrement abordable à l'échelle d'une équipe de recherche.

Depuis une décennie, les architectures parallèles et les architectures distribuées à base de fermes de calcul constituant des grilles de calcul sont en plein essor. Ces dernières consistent à faire collaborer plusieurs fermes afin d'augmenter la puissance de calcul totale.

Plusieurs technologies de programmation des architectures parallèles ont vu le jour depuis : les APIs[1] parallèles comme MPI[2] ou OpenMP[3] et les middlewares de grilles comme Globus[4] ou gLite[5].

Afin de contribuer à l'amélioration des performances des logiciels de conception de sondes pour biopuces à ADN pour l'environnement, les objectifs de la thèse étaient les suivants. Tout d'abord, nous avons souhaité apporter une simplification du point de vue logiciel en exploitant les méthodes de génie logiciel et de conception de systèmes d'information pour les appliquer à la sélection de sondes pour les biopuces à ADN. Ensuite, nous avons voulu exploiter les outils bioinformatiques utilisés dans la sélection de sondes *in silico* et tirer profit de leur parallélisation sur différentes architectures matérielles. Enfin, nous avons développé des outils permettant de déployer à petite et à grande échelle des logiciels de conception de sondes sur architectures parallèles et notamment sur grille de calcul.

Cette thèse s'est déroulée conjointement dans les laboratoires LIMOS et LMGE de l'université Blaise Pascal à Clermont-Ferrand sous la direction des professeurs David HILL et Pierre PEYRET. Le mémoire de cette thèse est présenté en quatre grands chapitres.

Le premier chapitre présente le contexte général de la thèse ainsi que la problématique biologique impliquant les développements entrepris. Tout d'abord nous présentons la biodiversité microbienne et l'intérêt de son étude. Ensuite nous introduisons les technologies de biopuces à ADN comme outils haut débit pour la découverte et la compréhension de la biodiversité. Enfin, nous parlons des problèmes posés par l'augmentation des données génomiques et les réponses que peuvent apporter l'informatique et plus précisément les architectures parallèles pour le problème de sélection de sondes.

Le deuxième chapitre propose un état de l'art et une synthèse bibliographique sur la sélection de sondes pour les biopuces à ADN. Les critères de sélection de sondes ainsi qu'une comparaison des logiciels décrits dans la littérature sont présentés. Ensuite, nous évoquons les avancées du point de vue du génie logiciel en étudiant notamment l'ingénierie dirigée par les

[1]API: Application Programming Interface
[2]MPI: Message Passing Interface
[3]OpenMP: Portable Shared Memory Parallel Programming
[4]Globus: Projet de développent d'un logiciel de Grille de calcul
[5]gLite : Middleware de Grille de calcul

modèles en tant que démarche efficace pour la conception de logiciels. Enfin, nous détaillons les technologies de calcul parallèle existantes, depuis les multiprocesseurs jusqu'aux fermes de calculs puis aux grilles de calcul.

Le troisième chapitre présente les approches logicielles proposées pour apporter des solutions concrètes aux verrous technologiques rencontrés. Dans un premier temps, un système d'information pour la centralisation des données génomiques est présenté. Ensuite, une approche de méta-programmation en ingénierie dirigée par les modèles est proposée pour augmenter l'aspect exploratoire des sondes utilisées pour les biopuces métaboliques. Enfin, une utilisation en parallèle sur grille de calcul de l'un des logiciels les plus populaires dans la génomique et la bioinformatique, BLAST (Basic Local Alignement Search Tool), est présentée.

Dans le dernier chapitre, nous présentons dans un premier temps la parallélisation sur grille de calcul de l'un de nos logiciels développé ainsi que les améliorations apportées. Dans un deuxième temps, nous présentons un nouveau logiciel de conception de sondes ADN pour biopuces métaboliques exploitant un nouveau système d'information en suivant une démarche d'ingénierie logicielle. Enfin, une conclusion générale et les perspectives de travaux ultérieurs clôtureront le mémoire.

Chapitre I : Contexte et problématique biologique

1. Introduction

1.1. Définitions et généralités

Le terme « biodiversité » ou « diversité biologique » a été utilisé pour la première fois dans les années 80 et a été défini comme étant la variété et la diversité du monde vivant. La biodiversité reflète le nombre, la diversité et la variabilité des organismes vivants, ainsi que la façon dont ces aspects changent d'un endroit à l'autre et avec le temps. Les biologistes ont continué à chercher une définition universelle jusqu'au milieu des années 90. Certains comme Adams ont considéré que le terme est dorénavant largement employé dans la littérature et n'admet pas une définition unifiée (DeLong, 1996). Parmi les définitions de la « diversité biologique », celle donnée par Schwarz comme étant « toute la diversité et la variabilité de la nature » et celle donnée par Spellerberg et Hardes lors du National Forum on Biological Diversity en 1986 avec l'emploi du terme définitif « biodiversité » comme étant « la variété de la nature et de ses processus » sont les plus significatives. L'intérêt porté à la biodiversité a augmenté de manière quasi-exponentielle durant les années 80 et 90 (Oksanen et al., 2004) et a coïncidé avec la prise de conscience mondiale de la disparition d'espèces vivantes et la nécessité de protéger et de préserver la nature. Il existe 3 niveaux de biodiversité :

- La biodiversité génétique qui consiste à décrire la diversité des gènes au sein d'une même espèce (groupe d'êtres vivants interféconds). Il s'agit de la diversité intra-spécifique.
- La biodiversité spécifique qui décrit la diversité des espèces déterminée par la taxinomie. Il s'agit de la diversité interspécifique.
- La biodiversité des écosystèmes qui décrit la diversité des écosystèmes de la Terre.

L'homme, faisant partie de la nature, plus au moins conscient du rôle des autres organismes pour sa propre survie, a toujours essayé de catégoriser le monde du vivant. Depuis l'utilisation de caractères de similarité morphologique jusqu'à la description de la diversité génétique, les classifications ne cessent d'évoluer. L'interdépendance de tous les organismes assurant le bon fonctionnement des différents écosystèmes a conduit ces dernières années à une prise de conscience de la nécessité de préserver la biodiversité.

Cinq évènements paroxysmaux d'extinction d'espèces ont rythmé l'histoire de la Terre. Plusieurs signes alarmants (modifications d'usage des écosystèmes, démographie humaine,

surexploitation des ressources vivantes, changement climatique actuel) indiquent qu'une 6éme extinction de masse est probablement en cours. Les tentatives d'estimation de l'érosion de la biodiversité restent cependant très imprécises.

Afin d'évaluer cette érosion, il est nécessaire de mesurer et d'estimer la biodiversité en développant de nouvelles approches qualitatives et quantitatives (Sarkar, 2005). En conséquence, pour mieux cerner la biodiversité au travers de la caractérisation des espèces, de nombreux paramètres doivent être considérés tels que la définition des espèces (diversité spécifique), l'abondance de ces espèces, la diversité fonctionnelle (définition des services écosystématiques), la variation individuelle définissant la richesse infraspécifique et la distribution dans l'espace faisant référence à la notion de biogéographie. Signalons que l'estimation de la biodiversité est souvent confondue avec l'estimation de la diversité d'espèces car les méthodes théoriques utilisées sont souvent basées sur des indices relatifs à l'abondance d'une part et à la richesse spécifique d'autre part (Hamilton, 2005). Malgré l'efficacité – qui reste relative – des mesures prenant en compte uniquement la richesse des espèces, il est également conseillé d'intégrer les paramètres écologiques et environnementaux qui interviennent souvent dans les prises de décisions politiques à l'échelle mondiale (Gotelli et al., 2001). Les indices les plus souvent évoqués sont la caractérisation de l'état de santé de l'écosystème, le capital écologique, la performance écologique, les impacts sur le changement climatique (température moyenne globale et indice de concentration atmosphérique du CO2, premier gaz à effet de serre). D'après le rapport sur la biodiversité du Programme de l'Environnement des Nations Unies UNEP[1] (United Nations Environment Programme) du Centre de surveillance de la conservation du monde (World Conservation Monitoring Center), la diversité biologique serait de l'ordre de plusieurs millions d'espèces dont la grande majorité est encore inconnue. En d'autres termes, l'introduction, ainsi que le succès du terme « biodiversité » ont marqué plusieurs changements intervenus récemment dans notre perception et notre compréhension de la diversité du vivant. La diversité biologique présente deux aspects de ces changements. La première est l'ampleur insoupçonnée de la diversité des espèces. Les méthodes indirectes laissent à penser qu'il existe plus de 10 millions d'espèces animales, végétales et d'invertébrés de très petite taille. S'ajoute à cela une richesse microbienne inégalée. Après trois siècles de travaux sur la systématique (Science de la classification des êtres vivants), les scientifiques ont déjà énuméré 1,7 millions d'espèces et continuent à décrire de nouvelles espèces à un rythme d'environ 10 000 par an. Cette complexité du monde vivant nous oblige à repenser les stratégies d'exploration de la biodiversité et à déterminer de nouveaux indicateurs prédictifs. Au-delà de cet inventaire, les systématiciens ont utilisé l'approche cladistique élaborée par William Henning permettant de développer de nouvelles méthodes qui

[1]UNEP: http://www.unep.org/

rendent possible une véritable classification phylogénétique du vivant c'est-à-dire l'établissement d'une parenté évolutive entre les espèces. Ces méthodes ont radicalement changé la classification des organismes et, plus globalement, notre vision de « l'arbre de la vie ». Le deuxième aspect des changements dans notre perception de la diversité biologique est l'existence d'autres niveaux d'organisation que la diversité des espèces, des niveaux qui doivent être étudiés pour comprendre et mieux gérer la biodiversité (Chevassus-au-Louis, 2007).

Tout d'abord, la diversité intra-spécifique, jusqu'alors décrite principalement pour les espèces domestiques, est maintenant décrite pour toutes les espèces par l'utilisation de marqueurs moléculaires et sera bientôt l'objet d'une étude de masse du fait des développements récents des techniques de séquençage ultra-haut débit. De même, au niveau supra-spécifique, la diversité des communautés écologiques et des écosystèmes exige de comprendre le rôle des interactions entre organismes et avec leur environnement assurant le fonctionnement des écosystèmes. Les « points chauds » de la diversité sont particulièrement étudiés car ils pourraient, de plus, constituer des refuges d'espèces. Ainsi, une meilleure connaissance de la biodiversité contribuera à définir son rôle dans la stabilité et dans la résilience des écosystèmes naturels et anthropisés. Il sera alors possible de mieux quantifier l'érosion de la biodiversité et d'envisager des actions correctrices de préservation.

1.2. Types de mesure de la diversité

D'après Hawksworth (1995), mesurer la biodiversité en calculant le nombre d'espèces dans un site donné n'est pas suffisant, car elle doit exprimer surtout la différence entre les individus habitant ce site. Cependant, trois approches sont possibles pour mesurer la diversité d'une autre manière :

- Mesures taxonomiques : basées sur l'étude d'un groupe taxonomique supérieur à l'espèce, tels que le genre ou la famille. Les études montrent que la mesure de la biodiversité chez certains groupes est significative lorsqu'on mesure l'espèce mais reste plus efficace chez d'autres groupes d'individus (supérieurs du point de vue taxonomique).
- Mesures moléculaires : basées sur les caractères moléculaires des différents individus étudiés. Le pourcentage de similarité des acides nucléiques est un moyen moléculaire pour calculer la biodiversité. Mais malgré la fiabilité de cette approche, elle reste dépendante de la nature du groupe étudié (eucaryotes, procaryotes) qui a souvent des caractères spécifiques qui évoluent différemment entre les espèces qui le compose.

- Mesures phylogénétiques : basée sur des méthodes de reconstruction phylogénétique faisant intervenir la distance taxonomique et/ou la théorie de l'évolution. Cette approche a pour but de calculer le niveau de conservation d'une espèce dans un site donné. Elle fournit un indice de biodiversité à différents niveaux taxonomiques (royaume, phylum, …), mais elle dépend des données disponibles pour l'analyse taxonomique.

Inopinément, après la découverte des microorganismes, il y a eu un bouleversement de l'appréhension du monde du vivant. En effet, l'Homme a découvert la vaste richesse de la planète apportée par ces êtres vivants microscopiques. Depuis près de 4 milliards d'années, ils ont évolué de manière continue. Aujourd'hui, on estime à seulement 1% le pourcentage des micro-organismes connus sur Terre, sachant que ces êtres vivants représentent le tiers de sa biomasse (Whitman et al, 1998).

2. Evaluation de la diversité microbienne

2.1. Nature de la diversité bactérienne

La microbiologie environnementale est la discipline permettant d'étudier la composition et la physiologie des communautés microbiennes dans l'environnement (écosystèmes terrestres, aquatiques, atmosphériques). La vie microbienne est présente dans tous les environnements même les plus extrêmes. Ils peuvent supporter une étendue de conditions physiques et chimiques qu'aucun autre organisme ne tolère (extrêmophiles) comme par exemple des températures au dessus de 100°C et des pH en dessous de 1 (Madigan et al., 1997 ; Fujiwara, 2002 ; Turner et al., 2007).

A l'opposé, on a trouvé en Antarctique des bactéries adaptées aux froids extrêmes, à la dessiccation et aux UV. On peut citer par exemple les communautés bactériennes qui poussent dans les particules minérales à 0°C dans les glaces permanentes des lacs antarctiques (Priscu et al, 1998).

Les microorganismes sont des acteurs clefs des grands cycles biogéochimiques et vivent en association avec de nombreux organismes dont l'Homme. Seule une minorité d'entre eux sont des pathogènes. Les capacités métaboliques des micro-organismes sont très diversifiées ce qui explique leur large utilisation en biotechnologies (production d'antibiotiques, d'acides, de protéines…). Dans l'environnement ils participent à la résilience d'environnements contaminés.

Leurs capacités de biodégradation sont donc utilisées dans des stratégies de bioremédiation de nombreux polluants dont les hydrocarbures, et les pesticides pour les plus connus. Les travaux de recherche menés depuis quelques décennies visent à caractériser les micro-organismes présents dans l'environnement, mais une attention particulière est apportée à ceux qui ont des capacités métaboliques permettant la restauration des écosystèmes.

Les études phylogénétiques moléculaires ont montré que la diversité microbienne et présente dans les trois grands royaumes du monde vivant : les Archées, les Bactéries et les Eucaryotes (Figure 1: Arbre phylogénétique des microorganismes (Maeva Kojta).). L'importance des microorganismes dans l'équilibre écologique de notre planète a longtemps été sous-estimée jusqu'à l'avènement des approches de biologie moléculaire permettant de révéler l'incroyable diversité microbienne (Hobbie et al, 1977).

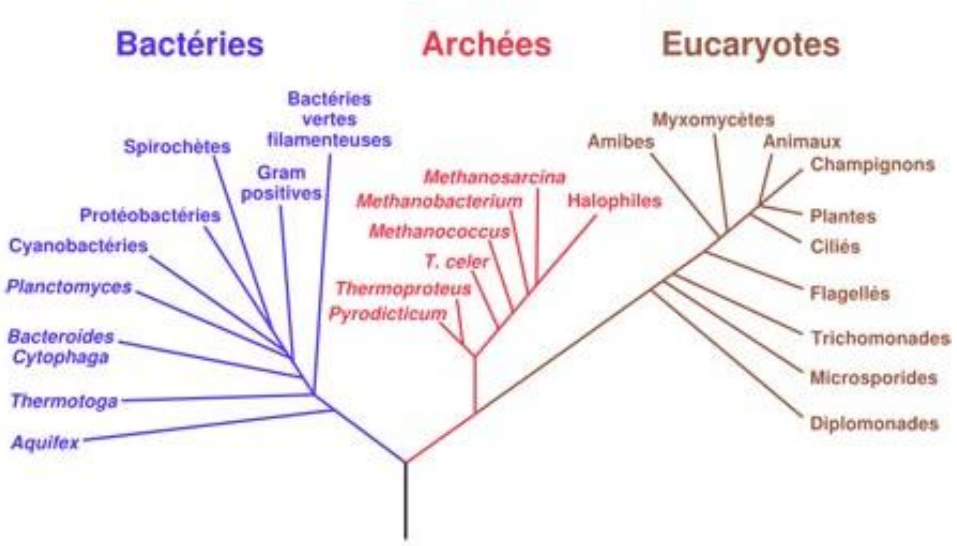


Figure 1: Arbre phylogénétique des microorganismes (Maeva Kojta).

L'anomalie de comptage en relation avec le faible nombre des bactéries cultivés sur boites de Petri a révélé qu'une très faible proportion des bactéries pouvait être cultivée par les méthodes actuellement développées (Staley et Konopka, 1985). Ces expériences ont démontré que n'importe quel millilitre d'eau douce ou d'eau de mer contient à peu près 10^5 à 10^6 bactéries. Au niveau terrestre un gramme de sol renferme près d'un milliard de bactéries. Même à plusieurs kilomètres sous terre des micro-organismes ont été découverts. Le nombre total de bactéries sur terre serait de l'ordre de 10^{30} selon certaines estimations. Les bactéries constituent sans doute près de la moitié du carbone organique sur terre et près de 90% de l'azote et du phosphore contenus dans les cellules vivantes (Whitman et al, 1998). Ils sont les acteurs principaux du recyclage de la matière organique et participent activement aux grands cycles biogéochimiques. De plus certains procaryotes, appartenant au royaume des Archées, sont responsables de la majeure partie des émissions de méthane qui entre dans l'atmosphère. Le méthane étant le second gaz à effet de serre derrière le CO2, ils contribuent au réchauffement climatique et à tous les problèmes environnementaux qui en découlent. Notons que les bactéries et les organites qui en dérivent (mitochondries et chloroplastes) sont les seuls à pouvoir extraire de l'énergie du rayonnement solaire (photosynthèse) et par oxydoréduction de la matière inorganique (lithotrophes). L'évaluation de la diversité microbienne reste complexe (Willey et al, 2007). Plusieurs approches on tenté d'effectuer cette évaluation (Smith et al., 2006) :

- **La diversité morphologique** s'attache à différencier les microbes suivant leur aspect externe (coques, bacilles, spirale, filament…). Leur dimension peut également varier de moins de 1 µm à plusieurs µm. Les cellules peuvent aussi être organisées suivant plusieurs modes d'associations (isolés, en paires, en chaînettes, en grappes…). Plusieurs espèces

peuvent s'associer pour former des matrices multi-espèces très structurées : les bio-films. Cette diversité morphologique ne permet pas de décrire la diversité microbienne.

- **La diversité structurale** s'attachera à mettre en évidence des structures particulières comme la paroi (coloration de Gram), les flagelles, les pilis, les fimbriae, les capsules ... Là encore cette diversité structurale n'est pas suffisante pour rendre compte de la diversité totale du monde microbien.
- **La diversité métabolique** repose sur la mise en évidence de capacités métaboliques particulières (type respiratoire, dégradation de molécules carbonés, dégradation de molécules azotées, systèmes de détoxication…). Même si ces approches sont plus discriminantes en combinant plusieurs caractères (galeries Api) avec la possibilité d'utiliser des systèmes automatisés (Biolog par exemple) elles sont encore insuffisantes pour rendre compte de la diversité microbienne.
- **La diversité génétique** s'attachera à décrire des différences de séquences de gènes jouant le rôle de biomarqueurs. Le gène codant pour la petite sous unité de l'ARN ribosomique est très largement utilisé en microbiologie comme nous le décrirons dans le chapitre suivant.

2.2. Méthodes d'évaluation de la diversité

Whitman et al., (1998) ont estimé que l'ensemble des procaryotes pourraient représenter jusqu'à $6\ 10^{30}$ cellules au niveau des écosystèmes terrestres et aquatiques faisant d'eux la plus importante biomasse portée par la Terre. En outre, ces microorganismes présentent une grande diversité (Curtis et Sloan, 2004) et une flexibilité métabolique et morphologique leur permettant de s'adapter à de nombreuses niches écologiques et aux variations environnementales naturelles ou accidentelles. Ces caractéristiques font qu'il est difficile de les étudier dans leurs milieux naturels. Cependant, leurs rôles dans le fonctionnement des écosystèmes et leur intérêt grandissant en biotechnologie nécessitent une meilleure connaissance de ces organismes.

Jusqu'à récemment, la mise en culture était une étape obligatoire dans l'identification et dans l'étude des microorganismes. Or, des travaux récents ont montré que seule une minorité de bactéries pouvait être isolée par les techniques de culture utilisées classiquement c'est-à-dire par les techniques d'enrichissement (Torsvik et Øvreås, 2002). Afin d'éliminer ce biais, il est alors apparu nécessaire aux microbiologistes de développer de nouvelles technologies leur permettant d'accéder à la majorité non cultivée des communautés bactériennes. Deux voies principales ont été empruntées afin d'atteindre cet objectif. La première a porté sur l'amélioration des techniques culturales avec le développement de nouveaux milieux de cultures et outils culturaux

(Ferrari et al., 2005 ; Joseph et al., 2003). Cependant, même si ces nouvelles techniques représentent un réel potentiel et une avancée considérable, elles s'avèrent encore inadaptées pour appréhender la totalité de la biodiversité des écosystèmes. La seconde a porté sur l'utilisation des approches moléculaires basées notamment sur l'analyse des séquences nucléiques des microorganismes. De plus, avec le développement de techniques moléculaires dites à haut-débit, la microbiologie exploratoire a pris un nouvel essor.

Durant la dernière décennie, l'utilisation de ces nouvelles techniques a connu un essor considérable, représentant une alternative puissante à l'identification des microorganismes par des approches culturales (Amann et al., 1995). Ces approches moléculaires incluent des techniques de réassociation d'ADN, d'hybridations ADN-ADN, de clonage-séquençage et d'empreintes génétiques comme par exemple la Denaturing Gradient Gel Electrophoresis (DGGE), la Temperature Gradient Gel Electrophoresis (TGGE), la Ribosomal Intergenic Spacer Analysis (RISA) et la Denaturing High-performance Liquid Chromatography (DHPLC) (Kirk et al., 2004 ; Nocker et al., 2007). Récemment, des techniques dites de métagénomique et de métatranscriptomique ont été développées afin d'accéder à l'immense réservoir génétique que forme l'ensemble des génomes et des transcriptomes de tous les microorganismes présents au niveau des environnements complexes. Ces méta « omics » peuvent permettre la découverte de nouveaux microorganismes et de nouvelles fonctions métaboliques grâce à la prise en considération globale de l'information génétique et non plus simplement de celle des biomarqueurs (Venter et al., 2004). Toutes ces techniques ont permis de faire des avancées considérables dans l'identification de la diversité phylogénétique et métabolique des communautés microbiennes présentes au niveau des écosystèmes. Actuellement, un des derniers défis que se sont lancés les microbiologistes consiste à établir un lien entre diversité et fonction afin d'accéder à l'écophysiologie des microorganismes. Afin d'atteindre cet objectif, différentes techniques, basées sur l'incorporation de marqueurs radioactifs (MAR-FISH, Isotope Array) ou isotopiques (Stable Isotope Probing) au niveau des macromolécules, ont été développées (Wagner et al., 2006).

Les biopuces ADN appliquées à l'étude des communautés microbiennes (DeSantis et al., 2007 ; Moisander et al., 2006 ; Peplies et al., 2006 ; Sanguin et al., 2006), est une autre approche représente un potentiel considérable car elle permet la détection simultanée de milliers d'espèces microbiennes en une seule manipulation (Wagner et al., 2007). Ainsi, l'utilisation de cet outil à haut débit pourrait permettre d'accéder à l'extraordinaire diversité microbienne d'environnements complexes et, plus précisément, d'étudier la dynamique des communautés bactériennes et de

mieux comprendre les mécanismes d'adaptation régissant le fonctionnement d'écosystèmes sains ou anthropisés.

Depuis le début des années 80, des efforts à l'échelle mondiale ont été entrepris pour sauvegarder les informations génomiques concernant tous les êtres vivants de notre planète. Une collaboration internationale des bases de données de séquences nucléiques a ainsi vu le jour en 1986.

3. Les données de génomique

3.1. Généralités

Les masses de données générées par les approches de génomique ont nécessité le développement d'outils de bioinformatique permettant d'exploiter efficacement les données biologiques et plus particulièrement les séquences nucléotidiques et protéiques. La bioinformatique est une discipline récente qui combine les mathématiques, les statistiques, et les technologies de l'information pour extraire de nouvelles connaissances à partir des données disponibles dans les banques de données. Aujourd'hui, près de 60 000 références bibliographiques sont répertoriées dans la base de données PubMed (gérée par le NCBI (National Center for Biotechnology Information) comportant le mot « bioinformatics », c'est dire l'importance de cette discipline qui date seulement des années 80.

3.2. Bases de données génomiques

Il existe 3 banques généralistes majeures répertoriant les séquences des acides nucléiques: l'EMBL (European Molecular Biology Laboratory) de l'EBI (European Bioinformatics Institute), GenBank (National Institue of Health genetic sequence database) et la DDBJ (DNA DataBank of Japan) [1]. Ces banques contiennent une description détaillée des données (fiches descriptives associées aux séquences nucléiques). Toutes les données sont échangées quotidiennement entre les groupes cités précédemment. Malgré une explosion des quantités de données génomiques, des efforts supplémentaires sont faits pour enregistrer les données provenant du séquençage à ultra haut-débit grâce aux nouvelles technologies (Cochrane et al, 2008). La taille de données stockées dans la base de données a augmenté de manière exponentielle depuis la création des banques de données (Figure 2).

Ces données sont organisées suivant une charte internationale permettant une compréhension universelle des différents éléments de description et d'annotation des séquences disponibles. Pour la banque de données EMBL, une table de référence est mise à disposition des chercheurs afin de faciliter la compréhension de la structuration de la base et des données. Les données génomiques stockées dans la banque de données EMBL sont définies suivant des classes de données préétablies. Elles décrivent généralement l'origine de la séquence enregistrée et la méthode qui a permis son obtention (Tableau 1).

[1] http://www.insdc.org/files/documents/feature_table.html

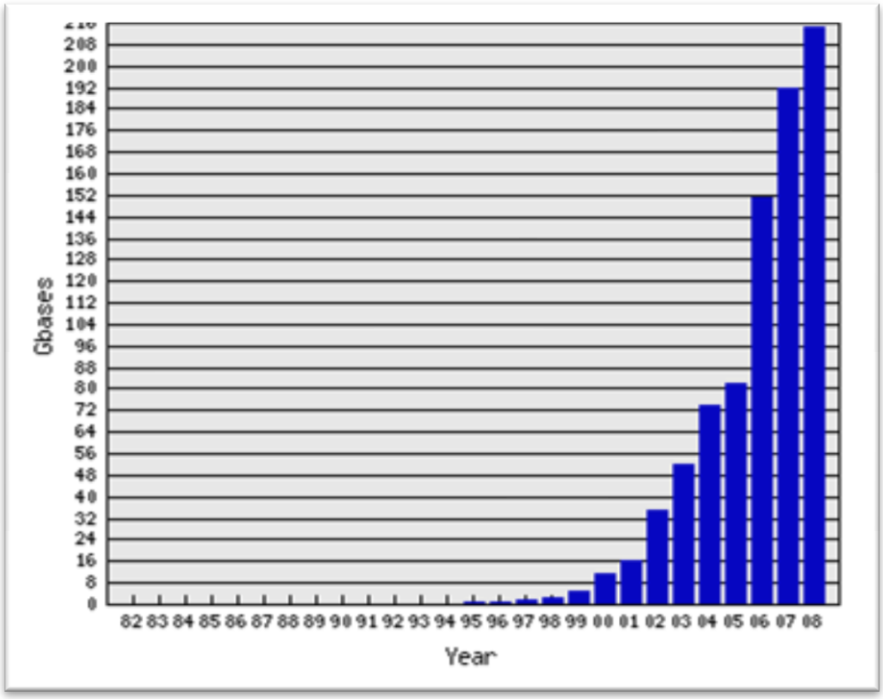


Figure 2: Nombre de nucléotides dans la base EMBL en 2008.

(Source: http://www.ebi.ac.uk/embl/Services/DBStats/)

Tableau 1: Types de classes de données EMBL.

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

Classe	Définition
CON	Entry constructed from segment entry sequences, drawing annotation from segment entries
ANN	Entry constructed from segment entry sequences with its own annotation
PAT	Patent
EST	Expressed Sequence Tag
GSS	Genome Survey Sequence
HTC	High Throughput CDNA sequencing
HTG	High Throughput Genome sequencing
MGA	Mass Genome Annotation
WGS	Whole Genome Shotgun
TPA	Third Party Annotation
STS	Sequence Tagged Site
STD	Standard (all entries not classified as above)

Par ailleurs, des conventions ont été établies telles que l'orientation des séquences. Les séquences sont toujours présentées par convention dans le sens 5' – 3' et elles sont numérotées à

partir du début 5' de la séquence. Pour faciliter l'exploitation de ces informations les entrées ont été organisées sous forme de fiches (fichiers formatés) bien structurées. Les entrées EMBL sont tout d'abord groupées sous formes de divisions taxonomiques (Tableau 2). De plus, dans chaque fiche la classification taxonomique de l'organisme étudié est décrite complètement si elle est connue.

Tableau 2: Types de divisions taxonomiques des données EMBL.

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

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

Chaque séquence est représentée par un numéro d'accession unique permettant de l'identifier dans la banque de données. Par ailleurs, un format EMBL est utilisé pour représenter les données sous forme textuelle (Figure 3). Chaque ligne du fichier commence par un code déterminant le type de données que va contenir la ligne. Par exemple, si la ligne commence par OS, le reste de la ligne contiendra le nom de l'organisme auquel appartient la séquence. Il existe des lignes obligatoires pour toutes les fiches EMBL telles que ID (Identifiant de la séquence), AC (Accession number : numéro d'accession), OC (Classification de l'organisme), ou encore SQ (la séquence proprement dite). Chaque entrée se termine par un « // ». Les lignes FT (features : caractéristiques) peuvent contenir des clés qui représentent des caractéristiques de la molécule décrite. Chaque clé peut être définie par plusieurs qualifiants d'une fiches EMBL (les mots clés

« /organism », « /mol_type »,… par exemple dans la Figure 3). Toutes les clés sont détaillées sur le site officiel de l'EBI : http://www.ebi.ac.uk/embl/WebFeat/.

```
ID   DQ417694_1; parent: DQ417694
AC   DQ417694;
FT   source          1..579
FT                   /organism="Crassostrea gigas"
FT                   /organelle="mitochondrion"
FT                   /isolate="Rochelle1_1"
FT                   /mol_type="genomic DNA"
FT                   /country="France"
FT                   /lat_lon="46.23 N 1.27 E"
FT                   /isolation_source="Atlantic east coast"
FT                   /collection_date="2003"
FT                   /note="PCR_primers=fwd_name: LCO1490, rev_name: HCO2198"
FT                   /db_xref="taxon:29159"
SQ   Sequence   579 BP;
     gctgttcttg cgggaactag gtttaggtct cttattcgtt ggagacttta taaccctgga        60
     gctaagtttt tagaccccgt gacttataat gcagttgtaa ctaggcatgc gttggttatg       120
     attttttttct ttgttatacc tgtaataatt ggggggtttg gtaactggct tatccctttg       180
     atgcttctag tagcagacat gcaatttcct cgattaaatg catttagatt ttgagttttg       240
     ccagggtctc tttatcttat gcttatgtct aacattgtag aaaacggagt tggggcaggg       300
     tgaacaattt accctccttt atcaacttac tcttatcatg gagtttgtat agaccttgca       360
     attctaagcc ttcaccttgc tggtattagc tctattttca ggtcaattaa tttcatagta       420
     acgattagaa atatgcgatc tgttgggggc catttactag cactattccc ttgatctatt       480
     aaggttactt cattcttgct tttgactact ctcccagtgt tagctggagg tcttactata       540
     cttttgactg atcgtcattt taatacctct ttttttgac                              579
//
```

Figure 3: Exemple de fiche EMBL.

La quantité importante de données enregistrées dans les banques de données génomiques internationales représente un défi majeur dans la sélection de sondes pour les biopuces à ADN. En effet, il est nécessaire de gérer correctement à l'aide d'algorithmes efficaces toutes ses données afin d'obtenir des oligonucléotides spécifiques et sensibles. Dans les parties suivantes, nous évoqueront la difficulté de la détermination de sondes *in silico* nécessitant une attention particulière du point de vue performances et conception.

4. Détermination de sondes pour biopuces à ADN

4.1. Principe des biopuces à ADN

Le principe de fonctionnement de la biopuce repose sur la complémentarité des bases entre les deux brins de l'ADN. Physiquement, les puces à ADN sont des supports (lames de verre ou de silicium) sur lesquels sont régulièrement répartis des fragments d'ADN simple brin appelés « sondes ».

Initialement, les sondes désoxyribonucléotidiques étaient fixées sur des membranes de nylon ou de nitrocellulose (macroarray : macro réseau) et les cibles étaient marquées par de la radioactivité (type ^{32}P). Ce support, limité en terme de densité de sondes, a vite été abandonné au profit de lames de verre pré-activées par une chimie de surface permettant la fixation d'un très grand nombre de sondes (microarray : micro réseau). Les sondes radioactives, quant à elles, ont été remplacées par des sondes fluorescentes en raison de leur flexibilité d'utilisation (plusieurs fluorescences différentes permettant des détections simultanées) et de leur facilité de manipulation (absence des contraintes liées à la radioactivité). Les formats de la taille d'une lame de microscope sont actuellement les plus répandus et apportent ainsi de nombreux avantages comme la haute densité, la possibilité de fixation covalente des sondes sur le support solide, la diminution de bruit de fond et la possibilité d'hybrider simultanément au moins deux populations de cibles (Dharmadi and Gonzalez, 2004). Le choix entre les deux principaux types de biopuces ADN, *ex situ* ou *in situ*, sera dicté par la nature des sondes, la densité désirée et le coût (Dufva, 2005 ; Kawasaki, 2006).

La préparation des cibles consiste en un marquage permettant la reconnaissance par le système de détection des duplex sondes/cibles formés lors de l'hybridation (Figure 4). La fluorescence est largement utilisée avec notamment l'incorporation des cyanines (Cy3 et Cy5) qui présentent des spectres d'émission de fluorescence bien distincts (respectivement 580-645 nm et 670-735 nm) et entraînent peu de bruit de fond (Ehrenreich, 2006). Plusieurs systèmes de détection ont été développés en fonction du type de plateforme utilisée. Les images des biopuces peuvent être obtenues à l'aide d'un scanner confocal ou non confocal, d'un scanner détectant la lumière issue de réactions de chimioluminescence ou par détection électrochimique (Timlin, 2006 ; Ghindilis et al., 2007). La qualité d'un scanner dépendra de la résolution de lecture (généralement 5-10 μm), du nombre de fluorescences détectées, des possibilités de lire

simultanément plusieurs fluorescences, de la vitesse de lecture, des capacités de chargement et de lecture automatique des biopuces et des logiciels de traitement des images associés. Des lectures d'hybridations en temps réel sont également possibles (Marcy et al., 2008). Les images générées sont sauvegardées au format TIFF (Tagged Image File Format) sans perte d'information (les pixels ayant généralement une gamme dynamique d'intensité allant de 0 à 65 535). Ces images sont le reflet qualitatif et quantitatif des hybridations. L'analyse des images nécessite plusieurs traitements successifs pour l'exploitation des résultats (Ehrenreich, 2006).

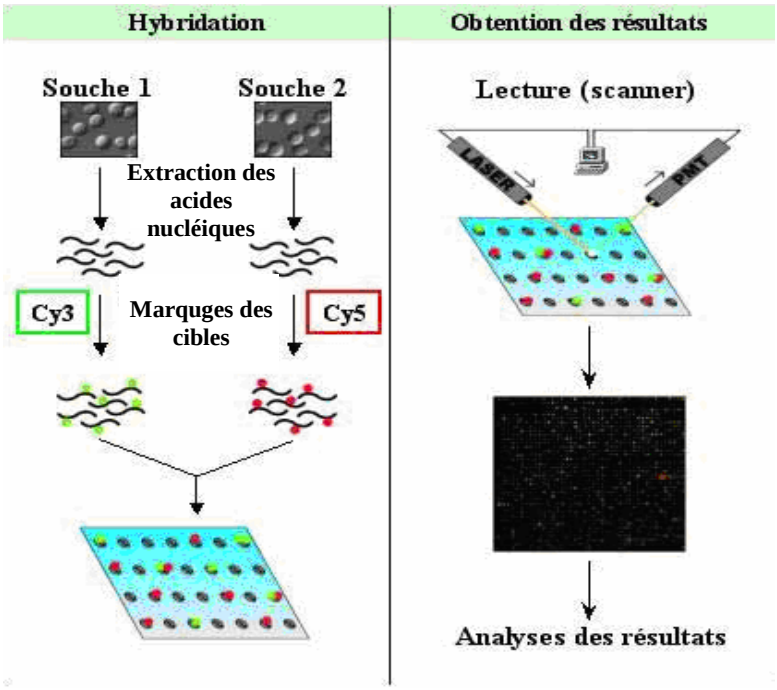


Figure 4: Principe de la biopuce à ADN.

4.2. Critères de sélection

La première étape qui intervient dans la conception d'une biopuce à ADN est la recherche de séquences qui jouent le rôle de véritables code-barres des molécules cibles à identifier (ARNm, ARNr, ADNg, ADNc, produits PCR) dans le milieu étudié. La sélection de sondes est donc une étape cruciale dans la conception de biopuces à ADN (Kreil et al, 2006). La recherche de séquences spécifiques doit être réalisée avec attention car plusieurs critères interviennent dans le choix des oligonucléotides :

- l'oligonucléotide ne doit pas s'hybrider sur lui-même (formation de dimères) et ne doit pas former de structures secondaires.
- la cible doit être facilement accessible.

- les hybridations croisées doivent être évitées afin de priviligier les hybridations spécifiques sondes/cibles.
- température de fusion homogène pour tous les oligonucléotides permettant de réaliser des hybridations à une même température (Stekel, 2003).

La recherche de sondes spécifiques nécessite de faire appel à des comparaisons de séquences très lourdes en terme de temps de calcul. Des outils de comparaison de séquences ont été développés depuis une vingtaine d'années pour rechercher les similarités de séquences. Parmi les programmes les plus utilisés, BLAST (Altschul et al., 1990) occupe une position privilégiée. En dépit des efforts entrepris pour proposer d'autres outils d'alignements locaux et globaux (FASTA, SAM, HUMMER, KALIGN, T-COFFE, MUSCLE, …), la recherche de similarité reste une tâche qui nécessite des temps de calcul très importants (Wu and Tseng, 2005 ; Chaudhary et al., 2005 ; Datta and Ebedes, 2005).

4.3. Les différents types de biopuces pour l'écologie microbienne

De nombreux types de biopuces ADN sont utilisables en écologie microbienne (Zhou et Thompson, 2002 ; Cook et Sayler, 2003 ; Ehrenreich, 2006 ; Gentry et al., 2006 ; Loy et Bodrossy, 2006 ; Sessitsch et al., 2006 ; Wagner et al., 2007). Ces biopuces apportent des informations sur la structure des génomes, leurs parentés, les expressions géniques, les capacités métaboliques, la structure et la dynamique des communautés microbiennes.

Les biopuces phylogénétiques oligonucléotidiques ou « Phylogenetic Oligonucleotide Arrays » (POA), ciblant les ADNr, sont actuellement le type de biopuces le plus largement utilisé pour étudier la structure et la dynamique des communautés bactériennes (Guschin et al., 1997 ; Small et al., 2001). Ces biopuces ont été utilisées pour caractériser:

- 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)
- 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), des environnements déterminés comme par exemple un sol pollué par l'hexachlorohexane (Neufeld et al., 2006),

- L'entière diversité d'environnements complexes (Wilson et al., 2002 ; DeSantis et al., 2005 ; Brodie et al., 2006 ; Palmer et al., 2006 ; Huyghe et al., 2008 ; Militon et al., 2007).

La version haute densité de la PhyloChip, constituée de 506 944 sondes de 25 mers, permet maintenant la détection simultanée de près de 9 000 unités taxonomiques opérationnelles (Brodie et al., 2006 ; DeSantis et al., 2007 ; Brodie et al., 2007).

Les biopuces fonctionnelles permettent quant à elles d'appréhender les capacités métaboliques des communautés microbiennes dans des environnements complexes. Ces biopuces ciblant les gènes fonctionnels ou « Functional Gene Arrays » (FGA) ont surtout été utilisées pour l'étude des gènes impliqués dans les grands cycles biogéochimiques et dans les procédés de bioremédiation (Bodrossy et al., 2003 ; Rhee et al., 2004 ; Wu et al., 2004 ; Zhang et al., 2007).

La GeoChip composée de 24 243 oligonucléotides de 50 mers ciblant plus de 10 000 gènes impliqués dans les cycles du carbone, de l'azote, du soufre, du phosphore mais aussi dans la dégradation de contaminants organiques et dans la réduction et la résistance aux métaux est la biopuce fonctionnelle la plus aboutie (He et al., 2007).

Les biopuces à oligonucléotides sont très largement utilisées du fait de leur facilité de fabrication. Cependant la détermination des sondes n'est pas triviale et demande des capacités de calcul importantes.

5. Ressources informatiques et calcul haute performance

5.1. Intérêt pour la biologie moléculaire

La biologie moléculaire *in silico* est rapidement devenue un domaine de recherche très important ces dernières années. En raison de la quantité et de la complexité du monde vivant, il est inconcevable de se passer des ordinateurs pour analyser leurs génomes. L'un des volets très actifs de la bioinformatique est la comparaison des séquences biologiques qui a pour objectif de déterminer les niveaux de similarité. Cette comparaison est importante car elle permet notamment d'évaluer la parenté des séquences (Zomaya, 2006). Dans le cadre de la conception de sondes pour les biopuces à ADN, une utilisation intensive des programmes de comparaison de séquences est nécessaire pour évaluer les niveaux de spécificité des sondes. L'accumulation des données de génomique nécessite de disposer de puissances de calcul supplémentaires pour la réalisation de ces tâches.

5.2. Le calcul intensif

5.2.1. Le calcul intensif (ou haute-performance)

Le calcul intensif (ou High performance computing (HPC)) est un terme qui désigne l'utilisation de supercalculateurs, de fermes de calcul ou de grilles de calcul pour résoudre des problèmes complexes. Durant les 20 dernières années, la perception du HPC a changé considérablement pour inclure également la communication entre machines et la capacité de stockage et ainsi devenir un système qui intègre puissance de calcul, transfert de données haut-débit et capacité d'agréger la puissance de machines distribuées (Hawick, 1997). De plus, l'augmentation du nombre d'applications qui nécessitent du calcul à haute-performance a contribué à l'évolution du HPC, d'un système basé sur les multiprocesseurs vers un système basé sur des « multi-ordinateurs ».

Pour obtenir un HPC, il faut généralement réunir:

- Une grande puissance de calcul, obtenue par l'assemblage de centaines, voire de milliers de processeurs performants,
- Une capacité de mémoire et de stockage étendue et rapide,

- Un système d'interconnexion extrêmement performant permettant de distribuer travail et données sur chaque processeur,
- Une gestion efficace des entrées-sorties.

Le TOP500[1] est la liste des 500 systèmes les plus performants installés dans le monde. Cette liste est mise à jour tous les 6 mois. Les calculateurs du TOP500 atteignent des puissances situées entre 1 et plus de 100 teraflops[2]. Il y a 4 ans, il n'y avait pas de système à base de processeurs Intel dans ce classement, aujourd'hui il y en a 2/3 (50% Xeon, 50% Itanium). Notons aussi que dans ce classement, IBM est particulièrement présent avec 57 références classées parmi les 100 premières (6 sur les 10 premières). Aujourd'hui, à l'heure de la guerre économique, la puissance informatique est un outil de la compétition mondiale à tous les niveaux. Les champs d'action sont variés :

- Militaire: avec la simulation du fonctionnement de l'arme atomique,
- Scientifique: avec les grands challenges actuels (climat, génome, physique corpusculaire...)
- Economique: les industriels figurent désormais parmi les principaux utilisateurs de puissance de calcul.

Le HPC (High Performance Computing) n'est pas destiné à simplement établir des records à un instant donné. Le classement TOP500 des supercalculateurs n'est pas l'équivalent du Livre des Records. C'est un baromètre des capacités technologiques, économiques et politiques des différents acteurs.

5.2.2. Des supercalculateurs aux fermes de calcul

Plusieurs architectures ont été développées depuis les années 60, date de l'apparition des tous premiers supercalculateurs avec IBM 7030, le CDC 6600 ou encore le CDC 7600 de la société Control Data Corporation. Ces machines ont rapidement passé la barre des 100 Mflops (Flop = unité de mesure de la puissance de calcul des machine et correspond à une instruction). L'apparition d'architectures vectorielles (vecteurs de nombres réels dits en virgule flottante) à la place des machines scalaires (simples nombres réels dits en virgule flottante) pendant les années 70 a permis de réaliser des traitements sur plusieurs données simultanément. Au début des années 80, le premier supercalculateur à multiprocesseur vectoriel a vu le jour (Cray X-MP/4) délivrant

[1] www.top500.org
[2]Un teraflop est égal à mille milliards d'opérations "flottantes".
[2]Un Mflop est égal à un million de flops.

près de 1 Gflops. Au début des années 90, ce sont les machines parallèles à mémoire partagée SMP (Symmetric Multi-Processing) qui ont vu le jour avec une disparition progressive des anciennes architectures. En même temps, une nouvelle architecture fondée sur le principe de mémoire distribuée avec échange de messages a rapidement progressé permettant d'éviter a priori les problèmes de contention d'accès à une mémoire partagée par un grand nombre de processeurs (≥ 128). Cette architecture comprenant généralement un grand nombre d'unités arithmétiques indépendantes est désignée par le terme MPP (Massively Parallel Processing). En 1994, après l'apparition des architecture de type constellation (ensemble de systèmes homogènes reliés par un réseau local) ou SIMD (Single Instruction Multiple Data) qui consiste en plusieurs unités de calcul fonctionnant en parallèle, c'est l'émergence des architectures « Cluster HPC » qui se basent sur les travaux de recherche de Thomas Sterling et Don Becker du centre d'excellence de la NASA qui jettent les bases des fermes de calcul Beowulf (www.beowulf.org). La Figure 5 montre l'évolution des architectures du calcul intensif avec l'apparition en aval de l'architecture Cluster (ferme de calcul) qui, avec l'apparition des librairies de communication pour architectures parallèles, ouvre une nouvelle ère du calcul haute performance.

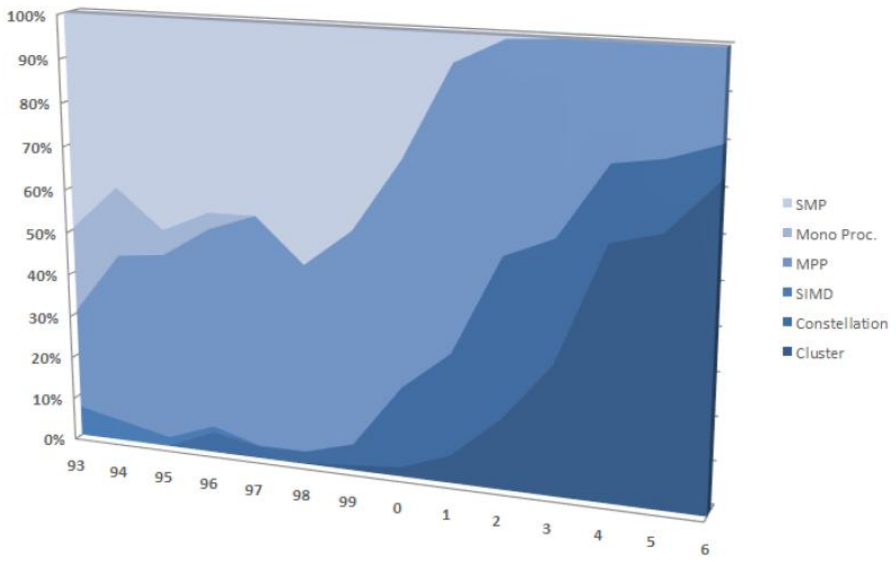


Figure 5: Pourcentage des différentes architectures sur le marché.

(Source : Le calcul intensif. High Performance Computing (HPC). Synthèse et prospective. http://www.microsoft.com/hpc/en/us/default.aspx).

Le principe ici est d'utiliser des ordinateurs classiques, fabriqués en grande série (architecture PC mono ou multiprocesseurs Intel ou AMD) donc peu onéreux sur lesquels seront répartis les traitements à effectuer. La synchronisation de ces traitements entre les différents nœuds de calcul se faisant par échange de messages via des réseaux dédiés haute performance et faible latence. Les programmes exécutés de façon parallélisée peuvent utiliser des bibliothèques de fonctions maintenant disponibles pour gérer les communications nécessaires entre les nœuds

d'une ferme de calcul telles que PVM (Parallel Virtual Machine) ou le standard de facto MPI14 (Message Passing Interface).

5.2.3. Des fermes de calculs aux grilles de calcul

La notion de parallélisme désigne le fait de pouvoir effectuer des calculs simultanément sur une architecture donnée. Cette notion s'est généralisée aux architectures parallèles de type cluster grâce au développement continu de middlewares (intergiciels) qui offre une performance croissante pour la résolution de problèmes de plus en plus complexes (Kacsuk et al, 2007). Les clusters représentent ainsi la première expérience consistant à réaliser un calcul simultanément sur différentes machines homogènes appartenant à un même réseau. Des architectures plus évoluées ont vu le jour très rapidement: il s'agit d'architectures distribuées qui comprennent plusieurs machines hétérogènes réparties dans l'espace et non forcément dédiées au calcul. On parle de « metacomputer ». Ce type de système s'est rapidement concrétisé sous forme de projets nationaux et internationaux ayant pour objectifs de profiter de la puissance de calcul qui existe dans chaque ordinateur dans le monde pour participer à la résolution de problèmes d'envergure. C'est à l'origine de telles avancées technologiques en informatique d'une part, et le besoin croissant exprimé par les scientifiques d'accéder à une puissance massive de calcul d'autre part, qu'est apparue la notion de grilles de calcul introduite pour la première fois en 1999 par Ian Foster et Carl Kesselman (Foster et Kesselman, 2004). Aujourd'hui, plusieurs dizaines de projets nationaux et internationaux sont recensés autour de la technologie des grilles au service de la recherche scientifique.

6. Problématique et conclusion

Dans cette partie nous avons présenté le cadre scientifique de la thèse en nous plaçant sous l'angle de la biodiversité et de l'écologie microbienne. Nous avons introduit les puces à ADN comme étant un outil de choix pour la détection des microorganismes à travers la conception de sondes spécifiques. L'objectif de la thèse est le développement d'algorithmes bioinformatiques pour la conception de biopuces ADN phylogénétiques et fonctionnelles en utilisant les dernières avancées du calcul sur grille. Nous avons vu, entre autre, que l'explosion des données biologiques disponibles dans les banques de données génomiques a poussé l'informatique à s'adapter en proposant de nouvelles méthodes de conception et à proposer des architectures destinées à utiliser le calcul parallèle et distribué pour augmenter la performance des logiciels bioinformatiques. La Grille de calcul représente le choix de l'avenir vu la démocratisation du séquençage massif et l'accumulation des données de séquences. Dans le chapitre suivant la conception de sondes pour les biopuces à ADN sera détaillée en décrivant tous ses aspects. Ensuite, le rôle de l'ingénierie dirigée par les modèles comme outil d'analyse informatique sera présenté. Enfin, le calcul parallèle et distribué fera l'objet d'une étude plus approfondie.

Chapitre II : Etat de l'art

1. La conception de sondes pour les biopuces à ADN

1.1. Définitions et généralités

Depuis quelques années, les puces à ADN sont devenues une technologie à haut-débit largement utilisée en biologie moléculaire. Ce sont les biopuces ADN à oligonucléotides qui sont les plus fréquemment rencontrées du fait de leur facilité de préparation par rapport aux biopuces à produits PCR par exemple. Leur utilité en microbiologie de l'environnement a été démontrée depuis une dizaine d'années (Wagner et al., 2007) et elles continuent à s'imposer comme une technologie de choix pour répondre aux questions d'écologie microbienne. Dans le contexte de la biodiversité microbienne des systèmes environnementaux complexes, le principe des biopuces à ADN est de détecter des gènes (présence et/ou expression) jouant le rôle de biomarqueurs. Les oligonucléotides constitueront les sondes qui seront fixées sur les supports solides formant la biopuce à ADN et seront capables de capturer les gènes cibles. Ces oligonucléotides doivent être choisis avec pertinence car ils représentent le point crucial de la conception des puces à ADN puisque ce sont eux qui vont déterminer la qualité du signal final après l'hybridation avec les cibles recherchées (Figure 6). Cette problématique de conception de sondes oligonucléotidiques pour les biopuces à ADN fait intervenir de nombreux critères influençant la spécificité et la sensibilité de détection. En effet, pour chaque cible (gène d'intérêt) une sonde oligonucléotidique optimale jouera le rôle d'un véritable code barre assurant la détection (Gasieniec et al., 2006). S'agissant d'échantillons environnementaux, les difficultés de conception des biopuces sont décuplées du fait de la présence de nombreuses espèces microbiennes dont la majorité reste non identifiée (Wagner et al., 2007 ; Bodrossy et al., 2004).

Ainsi, la qualité de l'hybridation entre les sondes des biopuces à ADN et les cibles du mélange biologique repose sur trois critères fondamentaux qui sont la spécificité, la sensibilité et la quantification (Zhou et al., 2002).

La spécificité est un critère représentant la notion d'une reconnaissance unique entre une sonde et une cible (Figure 6). Cette recherche de spécificité sera plus ou moins complexe selon le modèle d'étude biologique. Par exemple, une biopuce taxonomique de type POA (cf. section I.4.3) vise un même gène biomarqueur conservé (ADN ribosomique) présent chez tous les organismes rendant ainsi difficile la détection de régions spécifiques. La spécificité de la sonde

sera dépendante de sa longueur mais aussi du nombre, de la position et du type de mésappariements entre la sonde et les acides nucléiques non cibles.

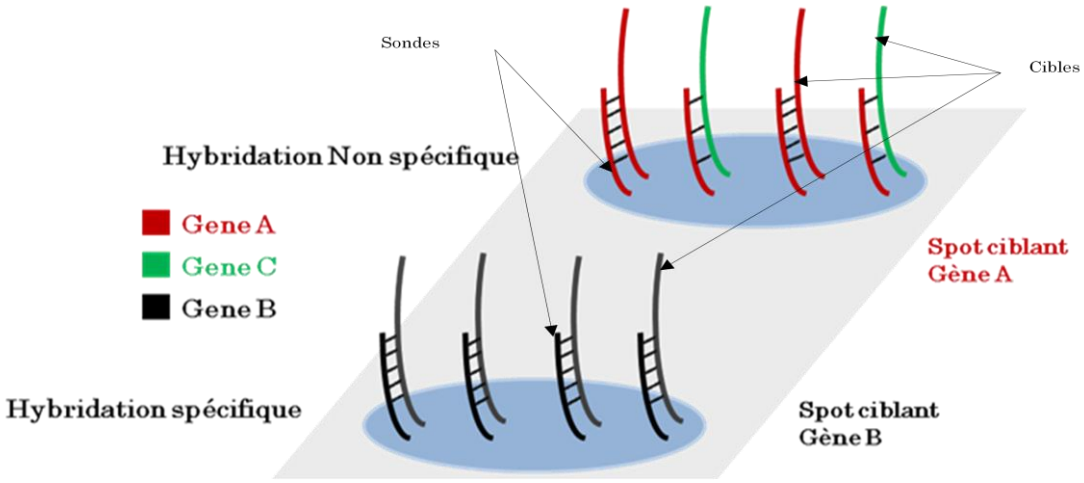


Figure 6: Représentation schématique d'une hybridation spécifique sonde-cible et d'une hybridation non spécifique (aspécifique).

La sensibilité est la capacité à détecter la présence de la cible même à faible concentration. Elle dépend de la longueur de la sonde mais aussi du type de support et de la méthode de fabrication de la biopuce (Small et al., 2001). La densité de sondes par spot et le type de liaison entre la sonde et le support solide de la biopuce influenceront les capacités de reconnaissance sonde-cible.

La quantification permettra d'évaluer le nombre de cibles présent dans l'environnement étudié (Taniguchia et al., 2001).

Dans le cadre de la conception de sondes oligonucléotides pour les puces à ADN, les caractères intrinsèques des séquences cibles sont essentiels pour la sélection *in silico*. (Hu et al., 2007). La force de l'approche biopuce ADN reposera donc sur sa capacité à détecter simultanément en une seule expérience des centaines de milliers de gènes différents.

1.2. Critères de sélection de sondes ADN

1.2.1. L'unicité

Souvent appelée spécificité de l'oligonucléotide (à ne pas confondre avec les principes d'hybridation (section II.1.1), par abus de langage, ce paramètre détermine les hybridations croisées potentielles avec des séquences non cibles du mélange biologique. Le problème d'hybridation croisée est sans doute le plus difficile à résoudre dans la conception des sondes à

ADN. En effet, plusieurs travaux de recherche ont été réalisés pour déterminer les effets d'une hybridation croisée sur l'intensité des signaux des spots (Wu et al., 2005 ; Zhang et al., 2004 ; Klebanov et al., 2007 ; Kachalo et al., 2004).

1.2.1.1. Recherche de similarité par alignement de séquences

Afin de détecter une possible hybridation croisée, il est nécessaire de pouvoir identifier les séquences présentant des similarités avec la sonde. En effet, un grand nombre d'algorithmes de recherche de similarité a été développé au cours des dernières années et utilisent tous des alignements de séquences pour en déduire un score qui correspond au degré de similarité entre la sonde et la cible. Le problème de détermination d'une sonde se résume à trouver un motif (séquence 1) dans un texte donné (séquence 2). Cependant, il faut également montrer que le motif n'est pas retrouvé dans une autre séquence avec un degré de similarité plus ou moins important selon les critères sélectionnés pour la conception de la sonde. La recherche exacte n'a donc pas d'intérêt dans ce cas. Il faut être capable de détecter des niveaux de similarité. Ainsi, ce problème est souvent basé sur la recherche de correspondance approximative entre deux séquences. Par conséquent, dans un tel modèle, le principe est de trouver à quel point deux séquences sont différentes. Le temps d'exécution de tels algorithmes de recherche peut varier d'un simple temps linéaire à un temps exponentiel. D'après Boukerche et al., (2006), le modèle mathématique est le suivant:

Quel que soit $s \in \sum^*$, où $\sum^*$ est un alphabet fini non vide, on note par $|s|$ sa longueur. Soit s_i le $i^{ème}$ caractère de s pour $i \in \{1 \ldots |s|\}$. On note aussi $s_{i,\ldots,j} = s_i\, s_{i+1} \ldots s_j$. Le problème de recherche approximative de motif se résume à :

Soit $|\sum| = \sigma$ est le nombre de lettre de l'alphabet $\sum$.

Soit $T \in \sum^*$ un texte de longueur $n = |T|$.

Soit $P \in \sum^*$ un texte de longueur $m = |P|$.

Soit $k \in \Re$ l'erreur maximale autorisée.

Soit $d : \sum^* \times \sum^* \rightarrow \Re$ la fonction *distance*.

Etant donné T, P, k, et d ( ), donner toutes les positions j telles que il existe i avec $d(P, T_{i,\ldots,j}) \leq k$. La distance $d(x, y)$ entre les chaînes de caractères x et y est le cout minimal d'un ensemble d'opérations de la forme $\delta(z, w) = t$, où z et w sont deux sous-chaînes de caractères différentes, qui transforme x en y comme l'approximation proposée dans (Navarro, 2001). Dans les applications de comparaison de séquences biologiques, les opérations δ possibles sont:

L'insertion $\delta(\varepsilon, a)$: insertion de la lettre a.

La délétion $\delta(a, \varepsilon)$: délétion de la lettre a.

La substitution $\delta(a, b)$: substitution de la lettre *a* par la lettre *b*.

L'alignement de deux séquences est obtenu par la transposition d'une séquence contre l'autre en observant les correspondances entre les caractères similaires. Des espaces sont insérés arbitrairement afin que les deux séquences aient la même longueur. Pour obtenir un score d'un alignement donné, trois cas de figure se présentent: un appariement (*match*), un mésappariement (*mismatch*), et un gap (Figure 7). Un appariement entre deux séquences *s* et *t* se réalise lorsque $s_i = t_j$, une valeur positive y est alors associée. Une valeur négative est donnée pour un mésappariement ($s_i \neq t_j$). Si s_i ou t_j est aligné avec un espace, un gap «_» se produit et une autre valeur négative lui est attribuée.

Figure 7: Alignement entre deux séquences avec présentation des scores.

Pour un appariement, on assigne +1, pour un mésappariement -1 et pour un gap -2. La somme est le score de l'alignement.

Les deux types d'alignement de séquences les plus largement utilisés sont:

L'alignement global obtenu lorsque la totalité de la longueur des séquences à aligner est considérée. Needleman et Wunch, (1970) ont proposé un algorithme basé sur la programmation dynamique pour calculer la matrice de similarité (ayant une complexité en $0(nm)$ où *m* et *n* sont les longueurs respectives des séquences) et la récupération de l'alignement local (ayant une complexité en $0(n)$). Sellers, (1974) a proposé un deuxième algorithme exact d'alignement de séquences quelques années plus tard.

L'alignement local obtenu lorsque seules des parties de séquences sont utilisées pour réaliser l'alignement. Smith et Waterman, (1981) ont proposé un algorithme d'alignement local

basé sur l'algorithme de Needleman en apportant quelques modifications sur la matrice de similarité.

1.2.1.2. L'algorithme de BLAST

Depuis le développement des ces premiers algorithmes, plusieurs autres algorithmes qui s'intéressent aux alignements de séquences ont vu le jour (Barton, 1998). L'alignement de séquences (recherche de similarité) est un outil très important dans la génomique comparative. Dans Boukerche et al., (2006), plusieurs algorithmes sont cités dont l'algorithme le plus ancien de Hirschberg, (1975). Le temps de calcul exponentiel de ces algorithmes a été à l'origine du développement d'algorithmes heuristiques conçus pour être plus rapides (Pearson et Lipman, 1988 ; Altschul et al., 1990). En effet, l'utilisation d'algorithmes exacts de recherche de similarité sur des banques de données de plus en plus importantes en termes de taille est rédhibitoire. Actuellement, BLAST (Basic Local Alignment Search Tool) (Altschul et al., 1990) est probablement le programme d'alignement local de séquences le plus largement utilisé. Il tient sa réputation de sa rapidité et la facilité de son utilisation (Casey, 2005). Paradoxalement, des dizaines de variantes de BLAST ont été développées depuis 1990 (PSIBLAST (Altschul et al., 1997) ; PHI-BLAST (Zhang et al., 1998) ; Mega-BLAST (Zhang et al., 2000) ; MPBLAST (Korf et Gish, 2000) ; BLASTZ (Schwartz et al., 2003)) sachant que la liste n'est pas exhaustive. L'algorithme de BLAST comporte 3 phases principales:

- **La recherche des mots voisins**: un mot est défini comme étant une série de lettres de longueur w apparaissant dans une séquence donnée. En effet, BLAST suppose que dans un alignement significatif les séquences ont des mots en commun. Dans un premier temps, toutes les positions des mots de longueurs w en commun entre deux séquences s et t sont déterminées par un appariement de motifs exacts. Seules les régions des séquences ayant des mots identiques seront utilisées comme origine 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 en lui-même et tous les autres mots ayant un score d'au moins T lorsqu'il est comparé à celui-ci par une matrice de substitution. C'est pourquoi le choix des paramètres w et T et de la matrice de substitution est important pour contrôler la performance et la sensibilité du BLAST.

- **Extension des mots**: les mots obtenus précédemment sont étendus pour générer un alignement en inspectant les caractères proches des mots dans les deux sens et en les concaténant progressivement jusqu'à l'obtention d'un score X appelé «drop-off » à partir duquel on estime qu'on ne peut plus réduire le score de l'alignement.

- **Evaluation**: les alignements générés dans la deuxième phase sont ensuite évalués pour enlever ceux qui sont non significatifs. En revanche, les significatifs appelés aussi HSP (High Score segment Pairs) sont ceux qui ont un score supérieur ou égal au seuil « *threshold* » *S*. De plus, des groupes de HSP sont générés afin de déterminer ceux qui sont les plus proches de la même diagonale. Les groupes de HSP consistants sont ensuite comparés au seuil final *E*, et seuls les alignements qui sont au-dessus du seuil sont considérés.

Les facteurs affectant la performance des recherches BLAST sont essentiellement la taille de la base de données associée à la recherche, la taille du fichier de séquences à comparer et la longueur de la séquence à comparer.

1.2.1.3. L'hybridation croisée

La principale utilisation d'un programme de comparaison de séquences consiste à rechercher les similarités entre les sondes potentielles et les cibles afin d'éviter les hybridations non spécifiques (hybridations croisées). En réalité, les hybridations croisées dans les biopuces à ADN pourraient être à l'origine de nombreuses erreurs de mesure de l'intensité des signaux d'hybridation. La complexité du problème de détermination de sondes est accentuée par l'impossibilité de prédire complètement le comportement de l'hybridation d'une sonde (Pozhitkov et al., 2007). Néanmoins, dans (Wu et al., 2005) il a été démontré que les hybridations croisées dans les puces à ADN sont majoritairement causées par des fragments d'une longueur de 10 à 16 mers complémentaires à la sonde (Nielsen et Knudsen, 2002). D'après plusieurs études (Evertzs et al., 2001 ; Xu et al., 2001 ; Wren et al., 2002), l'hybridation croisée dans l'approche biopuces à ADN est généralement constatée lorsqu'un oligonucléotide a une similarité de plus de 70 à 80%. Dans Kane et al., (2000), pour des oligonucléotides de 50 mers, une hybridation croisée est observée à partir d'une similarité de 75 à 80% avec un transcrit du mélange cible. De plus, les mêmes auteurs montrent que la présence d'une séquence identique continue de 15, 20 ou 35 bases sur une séquence non cible entraîne respectivement une intensité d'hybridation de 1%, 4% et 50% par rapport à une hybridation parfaite. D'après Kucho et al., (2004), pour obtenir des puces à ADN spécifiques et sensibles pour des sondes uniformes de 45-mer, les sondes doivent répondre aux critères suivants: une identité sonde-non cible inférieure à 71% et un pourcentage GC inférieur à 55%. Il est donc possible de distinguer deux types de similarités pour les hybridations croisées:

Type A: similarité pour des segments de séquence d'une certaine longueur minimale avec un pourcentage minimum d'identité (alignement long avec des mésappariements).

Type B: similarité pour des segments identiques d'une longueur minimale (appariement parfait court).

Le pourcentage de GC d'une sonde, ou encore le nombre de mésappariements et leurs positions entre la sonde et la cible peuvent aussi influencer la force des hybridations non spécifiques (Huang et al., 2005). Pour compenser l'hybridation croisée, les méthodes les plus utilisées dans la littérature (Huang et al., 2005) consistent à compter le nombre des mésappariements (*mismatch* MM) afin de soustraire leur contribution dans le signal global contenant les appariements parfaits (*perfect match* PM).

Les séquences à complexité réduite comportent des motifs répétés qui pourront être à l'origine d'hybridations croisées avec d'autres séquences. Plusieurs logiciels tels que REPEATMASKER, SIMPLE34, SEG, ou DUST ont été développés (Hancock et Armstrong, 1994 ; Wooton et Federhen, 1993 ; Wooton et Federhen, 1996 ; Altschul et al., 1994 ; Bedell et al., 2000 ; Morgulis et al., 2006) et permettent de substituer les régions répétées par des 'N' (ou des 'X') pour les séquences nucléiques (respectivement protéiques). Les oligonucléotides (1) et (2) représentent ici deux cas particuliers de séquences à complexité réduite susceptibles de provoquer des hybridations croisées.

(1) `TTTATTTTTTTTTTTTCGGAAGGTGCAGGAAT`
(2) `TATATATATATATATATATATAACACACGCAG`

La sonde (1) est composée d'une suite de T capable de s'hybrider avec des cibles portant des régions riches en A et la sonde (2) peut s'hybrider avec un large panel de séquences du mélange cible à cause des répétitions du motif TA. Ces deux exemples peuvent être analysés au début de la conception des sondes en modifiant à l'aide des algorithmes précédents les séquences de référence pour masquer cette complexité. Le résultat après traitement donne les séquences suivantes :

(1) `NNNNNNNNNNNNNNNNCGGAAGGTGCAGGAAT`
(2) `NNNNNNNNNNNNNNNNNNNNNNACACACGCAG`

Les séquences de faible complexité ne seront pas utilisées pour la détermination des sondes pour biopuces à ADN.

Les oligonucléotides courts sont plus spécifiques que les oligonucléotides longs car ils ciblent plus spécifiquement les séquences du mélange cible. En revanche, les oligonucléotides longs sont plus sensibles car ils permettent une détection même à faible concentration des séquences cibles. Une approche originale développée dans notre laboratoire permet de combiner les avantages de spécificité des sondes oligonucléotidiques courtes et les avantages de sensibilité des sondes oligonucléotidiques longues (Militon, et al., 2007; Rimour, et al., 2005). Cette approche permet de détecter deux séquences courtes spécifiques sur un gène cible afin de constituer un oligonucléotide long (concaténation des deux séquences courtes) augmentant la sensibilité.

D'une manière générale, hybrider de façon uniforme différentes sondes ciblant différents gènes en conservant un degré élevé de spécificité et de sensibilité à très haut débit reste un problème difficile à résoudre (He et al., 2005). D'après (Wagner et al., 2007), au-delà de la maîtrise des biopuces à ADN comme outil à haut-débit pour l'étude des microorganismes des environnements complexes, il est important de compléter les résultats de biopuces par des techniques de biologie moléculaire quantitatives telles que la FISH (Fluorescence *in situ* Hybridization) ou la qPCR par exemple. Cela est dû essentiellement à la difficulté rencontrée dans la recherche sans cesse d'une hybridation idéale pendant laquelle chaque sonde (parmi les milliers de sondes spotées sur la biopuce) s'hybride uniquement avec la cible pour laquelle elle a été conçue, ce qui est évidemment une hypothèse idéaliste et non réalisable concrètement. Notre contribution portera sur le développement d'algorithmes pour l'optimisation des sondes oligonucéotides pour les biopuces ADN.

L'obtention de sondes spécifiques et sensibles est un réel défi mais la sélection de sondes exploratoires en est un encore plus grand. En effet, seule une petite partie des communautés microbiennes est connue à l'heure actuelle, ce qui implique que la majorité des séquences des microorganismes n'est pas disponible dans les bases de données. La plupart des logiciels de sélection de sondes utilisent ces données incomplètes pour générer des sondes « spécifiques » d'espèces. Ainsi, seule la fraction connue des communautés microbiennes peut être étudiée à l'aide de ces sondes. Nos algorithmes sont conçus pour apporter cet esprit exploratoire permettant de détecter de nouveaux microorganismes et de nouveaux gènes.

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

La température de fusion (T_m) est la température à laquelle la moitié des molécules d'ADN sont sous forme dénaturée et l'autre moitié sous forme appariée. Dans le contexte des biopuces

ADN, il s'agit d'un critère permettant d'obtenir un ensemble de sondes montrant un comportement d'hybridation similaire pour une température d'hybridation donnée. Plusieurs méthodes ont été développées pour le calcul de la température de fusion.

Méthode d'approximation basique: Elle prend en compte uniquement le nombre de chaque base nucléique (Marmur et Doty, 1962). Elle est vérifiée pour des séquences d'une longueur maximale de 14 mers. Elle est donnée par la formule suivante:

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

Où wA, xT, yG et zC sont respectivement le nombre de A, T, G et C.

En revanche, la température de fusion est calculée pour des séquences d'une longueur supérieure à 14 mers par :

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

Dans les deux cas, on suppose que la concentration en sel et la concentration de l'oligonucléotide sont égales à 50 nM et que le pH est égal à 7.

Méthode d'ajustement: Elle repose sur l'intégration de la concentration en sel dans le calcul de la température de fusion. Suivant la longueur de la séquence, plusieurs équations sont utilisées dans la littérature. Howley et al., (1979) utilisent en partie les équations précédentes avec un ajustement par la concentration en sel. Elle est donnée pour une concentration en sel entre 0.01 et 0.4 M et un pourcentage GC entre 30% et 75% par:

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

Méthode thermodynamique: La thermodynamique des duplexes d'acides nucléiques peut être caractérisée selon les équations présentées ci-dessous. 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 de la perte de capacité du système à travailler (ou

le degré de perte de liberté du système), et, l'énergie libre (free energy) ΔG d'une réaction chimique est donnée par l'équation suivante:

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

Cette énergie libre mesure la stabilité de la réaction chimique à une température T_m et à une pression constante. Par définition, et en supposant que la réaction se fait dans les conditions réelles, la température de fusion d'un duplex est alors donnée par:

$$T_m = \frac{\Delta H}{\Delta S - R.\ln(\frac{C}{4})}$$

Avec *R* la constante molaire des gaz parfaits (égale à 1,987 cal mol^{-1}) et *C* la concentration molaire de la cible et *ln* le logarithme népérien, d'après Stekel, (2003).

Méthode du plus proche voisin: Elle prend en compte la relation entre l'entropie ΔS, l'enthalpie ΔH, la concentration en sel et la concentration de l'oligonucléotide (Wetmur, 1991). La formule est données par:

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

Cette formule suppose que l'hybridation est réalisée à un pH = 7, que la séquence contient au moins un C ou un G et que la longueur de l'oligonucléotide est comprise entre 14 et 20 nucléotides afin d'obtenir des températures de fusions raisonnables.

Enfin, d'après Panjkovich et Melo, (2005) « *des différences significatives sont observées pour les valeurs de Tm des oligonucléotides courts calculées par les différentes méthodes. D'autres données expérimentales portant sur un plus grand nombre d'oligonucléotides sont nécessaires afin d'évaluer l'exactitude des méthodes actuelles ou pour obtenir une estimation plus précise pour toute T_m expérimentale pour les courtes séquences oligonucléotidiques. Par conséquent, l'usage d'un calcul consensus de la T_m avec un minimum de probabilité d'erreur doit être proposé. Il devrait être dérivé de la comparaison des méthodes existantes à travers un grand nombre de séquences de référence. Les lignes directrices à suivre pour accroître le succès pratique de ces méthodes dans les applications de la biologie moléculaire sont les suivants :*

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

(2) Si possible, utilisez des séquences d'oligonucléotides ayant une composition moyenne en CG et qui sont plus courtes que 20 à 22 mers (où la plupart des méthodes de calcul de T_m sont applicables).

(3) Se référer à la littérature à venir pour des nouvelles méthodes plus précises pour le calcul de la T_m. »

La température de fusion est souvent associée à la formation de structures secondaires présentées dans la suite. Le choix de la température est donc crucial pour l'obtention d'une sonde optimale. Les propriétés thermodynamiques des duplex sondes/cibles, largement étudiées en solutions (SantaLucia, et al., 1996) ont aussi été utilisées pour sélectionner des sondes pour biopuces (Fotin, et al., 1998). En revanche, certains travaux suggèrent que les paramètres thermodynamiques ne peuvent pas être utilisés pour le design de sondes pour biopuces ADN car leurs influences sur les efficacités d'hybridations sur un support solide sont encore loin d'être entièrement connues (Pozhitkov, et al., 2007).

1.2.3. Les structures secondaires

On parle de structure secondaire lorsqu'une séquence nucléique présente des appariements entre certaines de ses sous-séquences (c'est-à-dire à l'intérieur du même brin). Cela concerne les sondes et les cibles. Une sonde, ou une cible, peut donc s'hybrider avec elle-même sous des conditions thermodynamiques bien précises. Cela peut empêcher que la sonde s'hybride complètement avec sa cible et peut donc fausser les résultats. Il est important alors d'éliminer ces oligonucléotides qui peuvent former des structures secondaires. Plusieurs méthodes de prédiction de structure secondaire ont été développées. Elles sont basées en grande partie sur les propriétés thermodynamiques des oligonucléotides (Stüber, 1986 ; Knudsen et Hein, 1999 ; Lyngso et al., 1999 ; Zuker, 2003 ; Knudsen et Hein, 2003 ; Ding et al., 2005 ; Do et al., 2006 ; Mathews, 2006 ; Wiese et Hendriks, 2006 ; Jabbari et al., 2008 ; Wiese et al., 2008). Lors de la sélection des sondes, il est donc important d'écarter les oligonucléotides qui s'hybrident sur eux même en formant une boucle comme dans la Figure 8 ou en s'hybridant entre eux sur le même spot. Pour cela, il faut identifier les oligonucléotides présentant des séquences de type palindrome (une séquence est un *palindrome* lorsqu'elle est identique à sa séquence inverse complémentaire). Le problème se complique davantage lorsqu'on sait qu'une séquence comme « ATTCG**GGTCG**ATTCG**CGACC**ATTCG » qui n'est pas un vrai palindrome mais qui est un palindrome partiel, puisse également former une structure tige-boucle.

Le problème de prédiction des structures secondaires peut être assimilé à un problème de minimisation de l'énergie libre d'une séquence simple brin par programmation dynamique

(Zuker, 2000). Il s'agit en fait de décomposer le problème global en cherchant une solution optimale pour chacun de ses sous-problèmes. Cela peut être vu de deux manières différentes: en résolvant le problème global en sauvegardant les solutions intermédiaires des sous-problèmes (Bottom-up) et remontant ainsi jusqu'au problème global ; ou en le résolvant de manière récursive en formulant un calcul complexe sous forme de séries de calculs plus simples (Top-down).

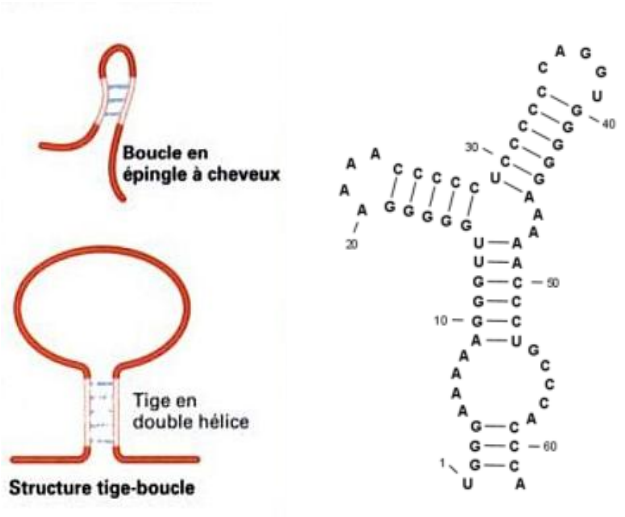


Figure 8: Exemples de structures secondaires des acides nucléiques.

Pour identifier par exemple une structure secondaire d'une séquence de longueur N ayant un maximum d'appariements entre ses bases nucléotidiques (un G avec un C, et, un T avec un A), le système de calcul de score se base simplement sur le principe suivant: +1 pour un appariement et 0 pour le reste. Si on considère une sous-séquence de i à j avec $1 \leq i < j \leq N$ et i, j sont des positions quelconques sur la séquences alors $[i, j]$ est une sous-séquence de la séquence de départ de longueur N. On construit alors de manière récursive le score en prenant en compte les principes suivants:

Si i et j forment une paire de base, alors on l'ajoute à la structure pour $i+1, j-1$.

Si i ne forme pas de paire de base, alors on l'ajoute à une structure pour $i+1, j$.

Si j ne forme pas de paire de base, alors on l'ajoute à une structure pour $i, j-1$.

Si i et j forment deux paires de bases avec deux autres positions, alors on les ajoute à deux sous-structures pour deux sous-séquences, $i \ldots k$ et $k+1 \ldots j$; où $i < k < j$.

La Figure 9 est extraite de (Eddy, 2004) et résume parfaitement l'algorithme de programmation dynamique par récursivité en utilisant la matrice de score.

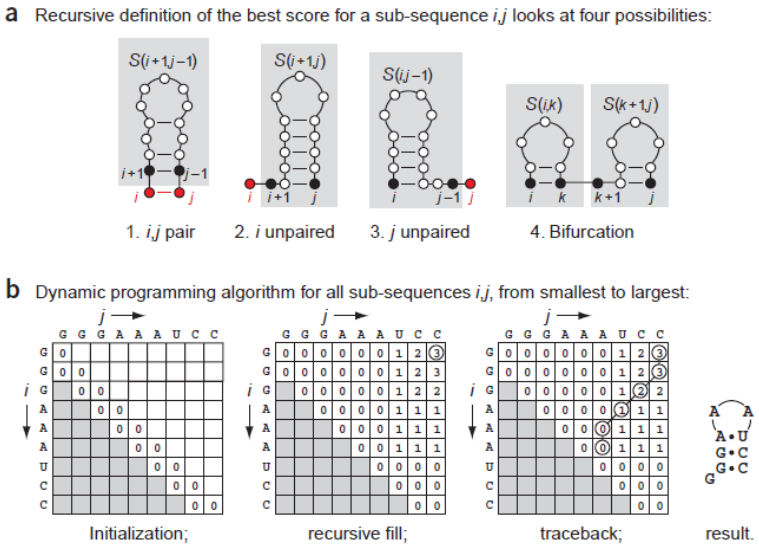


Figure 9: Programmation dynamique par récursivité tiré de (Eddy, 2004).

Pour la prédiction de structure secondaire, les algorithmes de recherche de la matrice des scores se résument à maximiser l'équation suivante :

$$S(i, j) = \max\{S(i+1, j+1) + 1 ; S(i+1, j) ; S(I, j-1) ; \max_{i<k<j} S(I, k)+S(k+1, j)\}$$

Malgré tous les efforts d'optimisation par des algorithmes heuristiques, les programmes de recherche de structures secondaires sont capables de trouver correctement seulement 50 à 70 % des paires de bases. C'est utile mais reste insuffisant pour obtenir les meilleurs résultats.

Les méthodes de recherche de structures secondaires peuvent être considérées comme une partie supplémentaire du processus de conception des sondes et pouvant intervenir à la fin de la recherche des sondes optimales pour les rendre encore plus fiables. Malgré les avancées dans ce domaine et la quantité des publications engendrées, toutes les approches emploient les mêmes mécanismes de l'algorithme de programmation dynamique sous-jacent proposé par Eddy, (2004).

Dans He et al., (2005), des critères empiriques ont été établis concernant par exemple le nombre de mésappariements autorisés et leurs positions sur la séquence pour une hybridation, l'énergie libre, la longueur de la séquence, la longueur d'un fragment identique à la cible (stretch), le pourcentage d'identité (relatif à la spécificité « unicité » section 1.2.1) ou encore le pourcentage des nucléotides G et C dans l'oligo. Afin de pouvoir automatiser la recherche de sondes optimales pour une séquence de référence, un nombre important d'algorithmes a été développé depuis quelques décennies pour répondre aux besoins de chaque domaine. Dans la partie suivante, nous décrierons les logiciels de conceptions de sondes ADN en complétant l'étude effectuée par Kreil et al., (2006).

1.3. Les algorithmes de conception de sondes oligonucléotidiques

1.3.1. Intégration des critères *in silico*

Les critères vus dans le chapitre précédent représentent fondamentalement le noyau dur des algorithmes de sélection de sondes oligonucléotides. En effet, les logiciels développés depuis l'invention des puces à ADN ont tous comme objectif de produire des oligonucléotides spécifiques et sensibles (cf. II.1.1). Tous ces logiciels ont pratiquement la même méthodologie globale pour l'intégration des critères de sélection. En effet, tout programme de design de sondes peut être assimilé à une suite d'étapes de calcul qui inclut plus ou moins des critères présentés dans la Figure 10 concernant les oligonucléotides. De plus, les opinions sur l'utilisation du programme BLAST pour le test de spécificité (unicité II.1.2.1) des sondes divergent. D'une part des scientifiques développent leurs propres outils de recherche de spécificité comme ProbeCheck (Loy et al., 2008), OligoCalc (Kibbe, 2007) ou ARB (Ludwig et al., 2004), et d'autre part une majorité fait confiance au programme BLAST car ce dernier donne une similarité entre la sonde et la cible sur une région prédéfinie par l'algorithme et qui ne permet pas d'obtenir une identité sur la totalité de la séquence non-cible.

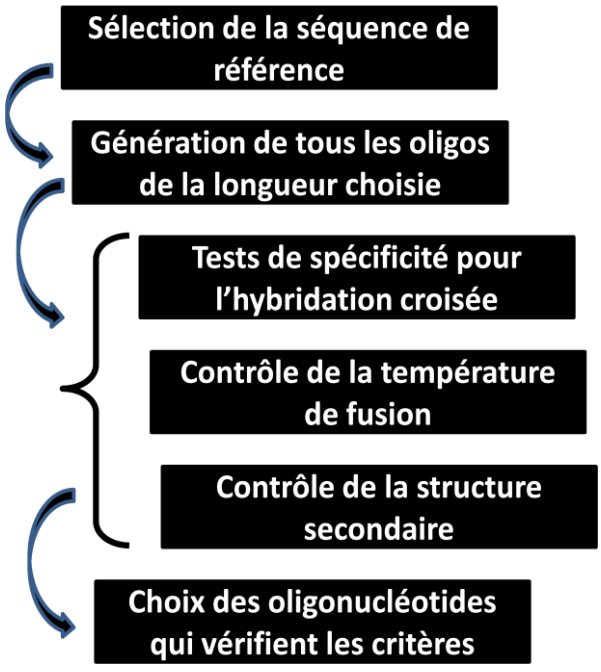


Figure 10: Schéma de conception de sondes. Il s'agit ici d'un algorithme typique permettant de visualiser les différentes étapes de conception. (Source: Sambrook et Russell, 2001)

En dépit de cette constatation, qui touche quand même le critère fondamental sur lequel repose la notion d'hybridation croisée, les logiciels disponibles dans la littérature utilisent majoritairement le programme BLAST ainsi que les critères de sélection « classiques » : le

pourcentage de similarité entre un oligonucléotide et la base de référence, la complexité, la température de fusion et la structure secondaire.

1.3.2. Les algorithmes de sélection de sondes les plus populaires dans la littérature

D'après Tomiuk et Hofmann, (2001), il est possible de distinguer plusieurs stratégies de sélection de sondes ADN suivant le type de problème auquel le chercheur est confronté. Dans le contexte des puces à ADN, il faut trouver un équilibre entre le débit des expériences, la précision des résultats (suivant que l'on utilise des oligonucléotides synthétiques ou de l'ADN complémentaire) et la nécessité de faire des efforts supplémentaires avant ou après l'hybridation.

La problématique de conception de sondes pour les biopuces à ADN est relativement récente. En effet, elle s'est développée manifestement lors de la dernière décennie. Le Tableau 3 récapitule les principaux algorithmes et logiciels disponibles gratuitement et qui ont fait l'objet d'une publication scientifique. Certains logiciels n'ont pas été intégrés dans la liste car ils font partie d'un package plus complet de traitement de séquences [ARB (Ludwig et al., 2004)] ou qui présente simplement le résultat d'une méthodologie de conception (Stoffels et al., 2001 ; Leparc et al., 2009) ou encore qu'ils n'ont pas d'URL qui fonctionne (Pozhitkov et Tautz., 2002). Les données peuvent évoluer au cours du temps car des nouvelles versions des logiciels peuvent être développées.

Tableau 3: Les logiciels de conceptions de sonde dans la littérature et leurs URL.

ArrayOligoSelector http://arrayoligosel.sourceforge.net/ **(Bozdech et al., 2003)**
CommOligo http://ieg.ou.edu/software.htm **(Li et al., 2005)**
DEODAS http://deodas.sourceforge.net/ **(Diedrich, 2002)**
HPD http://brcapp.kribb.re.kr/HPD/ **(Chung et al., 2005)**
GoArrays http://www.isima.fr/bioinfo/goarrays/ **(Rimour et al., 2005)**
Mprobe 2.0 http://www.biosun.org.cn/mprobe/ **(Li et Ying, 2006)**
OligoArray 2.1 http://berry.engin.umich.edu/oligoarray2_1/ **(Rouillard et al., 2003)**
OliCheck http://www.genomic.ch/techno_array.php **(Charbonnier et al., 2005)**
Oligodb http://oligodb.charite.de/ **(Mrowka et al., 2002)**

OligoDesign http://oligo.lnatools.com/expression/ **(Tolstrup et al., 2003)**
OligoFaktory http://ueg.ulb.ac.be/oligofaktory/ **(Schretter et Milinkovitch, 2005)**
OligoPicker http://pga.mgh.harvard.edu/oligopicker/index.html **(Wang et Seed, 2003)**
OligoRankPick http://zblab.sbs.ntu.edu.sg/OligoRankPick/ORP.tar.gz **(Hu et al., 2007)**
OligoWiz 2.0 http://www.cbs.dtu.dk/services/OligoWiz/ **(Nielsen et al., 2003; Wernersson et Nielsen, 2005)**
Oliz http://www.utmem.edu/pharmacology/otherlinks/oliz.html **(Chen et Sharp, 2002)**
Osprey http://osprey.ucalgary.ca/ **(Gordon et Sensen, 2004)**
Phylarray http://fc.isima.fr/~rimour/phylarray/ **(Militon et al., 2007)**
Picky http://www.complex.iastate.edu/download/Picky/tutorials.html **(Chou et al., 2004)**
PRIMEGENS http://compbio.ornl.gov/structure/primegens/ **(Xu, 2000)**
PRIMROSE contact : ashelford@cardiff.ac.uk **(Ashelford et al., 2002)**
ProbeMaker http://probemaker.sourceforge.net/ **(Stenberg et al., 2005)**
PROBEmer http://probemer.cs.loyola.edu **(Emrich et al., 2003)**
PROBER http://prober.cshl.edu **(Navin et al., 2006)**
PROBESEL http://www.zaik.uni-koeln.de/bioinformatik/arraydesign.html **(Kaderali et Schliep, 2002)**
ProDesign http://www.uhnres.utoronto.ca/labs/tillier/ProDesign/ProDesign.html **(Feng et Tillier, 2007)**
ProbeSelect - contacter : stormo@ural.wustl.edu **(Li et Stormo, 2001)**
ProkProbePicker contacter fox@uh.edu **(Zhu et al., 2006)**
Promide http://oligos.molgen.mpg.de/ **(Rahmann, 2003)**
ROSO http://pbil.univ-lyon1.fr/roso/ **(Reymond et al., 2004)**
SEPON http://www.agrsci.dk/hag/sepon/ **(Hornshøj et al., 2004)**
YODA http://pathport.vbi.vt.edu/YODA/ **(Nordberg, 2005)**

Afin d'avoir une vue globale sur les méthodes employées par les différents logiciels pour traiter le problème de spécificité (unicité des sondes), le Tableau 4 récapitule les approches

utilisées pour la détection d'hybridation croisée (cf. Partie II.1.2.1.3). Nous pouvons citer, par exemple, l'utilisation de programmes de recherche de similarité (BLAST, Suffix Array, EMBOSS ou SeqMatch) même si le programme BLAST reste le plus largement utilisé ; ou la recherche d'une portion de séquence constitué de nucléotides consécutifs identiques entre la sonde et des non-cibles (qui reste un critère pas toujours utilisé) ; ou la détermination du pourcentage d'identité (en se basant sur le résultat de BLAST ou en utilisant d'autres techniques plus triviales comme la comparaison base-à-base) ; ou encore la recherche des mésappariements possibles entre les cibles et la sonde pour éliminer celles qui peuvent s'apparier même en ayant des mésappariements à des positions pré-calculées avec des méthodes statistiques (Chung et al., 2005).

Tableau 4: Les composants des logiciels de conception d'oligonucléotides.

Logiciel	Programme de recherche de similarité	Bases identiques consécutive	Pourcentage d'identité	Mésappariement
ArrayOligoSelector	BLAST	Non	?	Non
CommOligo	BLAST	Oui	Oui	Oui
DEODAS	EMBOSS	Non	Non	Oui
GoArrays	BLAST, $w = 7$	Oui	Oui	Non
HPD	BLAST	Non	Oui	Oui
MProbe 2.0	BLAST	Oui	Oui	Non
OligoArray	BLAST, $w = 7$	?	Non	Non
Oligodb	BLAST	Non	Non	Non
OligoDesign	BLAST, $w = 9$	Non	Non	Oui
OligoFaktory	BLAST	?	Oui	?
OligoPicker	BLAST, w = 8	Oui	Non	Oui
OligoRankPick	BLAST	Non	Oui	Non
OligoWiz	BLAST	Oui	Oui	Non
Oliz	BLAST	Oui	Oui	Non
Osprey	BLAST	Oui	?	Non

Phylarray	BLAST	Oui	Oui	Non
Picky	Suffix array	Oui	Oui	Non
PRIMEGENS	BLAST	Non	?	?
PRIMROSE	?	Non	Oui	Oui
ProbeMaker	?	?	?	?
PROBEmer	Suffix Tree	Non	?	Non
PROBER	?	Oui	?	Non
PROBESEL	Suffix tree	Non	Oui	Non
ProDesign	Suffix Tree	Non	?	Non
ProbeSelect	Suffix array	Oui	Non	Non
ProkProbePicker	BLAST	?	?	?
Promide	Suffix array	Non	Non	Non
ROSO	BLAST, w = 7	Non	?	Oui
SEPON	BLAST	?	Oui	Non
YODA	SeqMatch, $w = 4$	Oui	Oui	Non

D'autre part, il est important de passer en revue les principaux critères d'élimination des mauvais oligonucléotides (Tableau 5). En effet, la spécificité et la sensibilité des sondes dépendent de différents paramètres comme:

Le pourcentage de GC: la fraction entre le nombre de bases G et C par la longueur L totale de l'oligo : (nG +nC/L)*100, où nG et nC sont le nombre des bases en G et C.

La température de fusion T_m: vue dans le chapitre II.1.2.3 et qui dépend suivant les formules employées du pourcentage GC.

Les structures secondaires: il est important d'éliminer les sondes qui peuvent former des structures secondaires à une température d'hybridation donnée.

La longueur: il est pratique de pouvoir choisir la longueur des sondes désirées. En jouant sur la longueur, on joue également sur la sensibilité et la spécificité des sondes (Rimour et al., 2005).

La complexité : afin d'éliminer les sondes ayant une forte complexité, certain logiciel font appel à des méthodes spécifiques (DUST).

Le type de cible : lors du test d'un logiciel de design de sondes le type de cibles qui sera caractérisé doit être pris en compte.

Tableau 5: Les principaux critères de conception de sondes des différents logiciels.

Logiciel	%GC	T_m	Structure secondaire	Longueur	Complexité	Type de cible
ArrayOligoSelector	Oui	Non	Oui (SW)	Non (70)	Oui (Lempel-Ziv)	ORF - Gènes
CommOligo	Oui	Oui	Oui (Self-annealing)	Oui	Non	?
DEODAS	Non	Non	Non	Oui	Non	Groupe
GoArrays	Non	Oui	Oui (DINAMelt)	Oui	Non	CDS
HPD	Oui	Oui	Oui	Oui (20-70)	Non	Groupe & Gènes
MProbe 2.0	Oui	Oui	Oui	Oui	Non	Gènes
OligoArray	Oui	Oui	Oui (Mfold)	Oui (45-47)	Non	Génomes
OliCheck	?	?	?	?	?	Génome
Oligodb	Oui	Oui	Oui (Mfold)	Oui	Non	Génome humain
OligoDesign	Non	Oui	Oui (Nussinov)	Oui	Oui	LNA
OligoFaktory	Oui	Oui	Oui	Oui	Non	Tout
OligoPicker	Oui	Oui	Oui (Blast)	Oui	Oui (DUST)	Protéines-CDS
OligoRankPick	Oui	Oui	Oui (SW)	Oui	Oui (Lempel-Ziv)	Génomes
OligoWiz	Non	Oui	Oui (Mfold)	Oui	Oui	Génomes
Oliz	Non	Non	Non	Non (50)	Non	?
Osprey	Oui	Oui	Oui (Mfold)	Oui	Oui (DUST)	Séquençage
Phylarray	Oui	Oui	Non	Oui	Non	Groupe ARN 16S
Picky	Oui	Oui	Oui	Oui	Non	Génomes
PRIMEGENS	?	Non	Oui (Primer3)	Oui	Oui	Génomes

PRIMROSE	Non	Non	Non	Non	Non	Groupe ARN 16S
ProbeMaker	Non	Oui	Non	Non	Non	?
PROBEmer	Oui	Oui	Oui (Mfold)	Oui	Non	Gene
PROBER	Oui	Oui	Non	Oui	Non	FISH
PROBESEL	Non	Oui	Oui (Mflod)	Oui	Oui	Tout
ProDesign	Oui	Oui	Oui	Oui (20-70)	Oui (Lempel-Ziv)	Gene & Groupe
ProbeSelect	Oui	Oui	Oui (Mfold)	Oui	Non	Gene
ProkProbePicker	?	?	?	?	?	Groupe ARN 16S
Promide	Non	Oui	Oui	Oui	Non	Tout
ROSO	Oui	Oui	Oui	Oui	Oui	PCR Tout
SEPON	Oui	Oui	Oui (Tm)	Oui	Non	EST Tout
YODA	Oui	Oui	Oui	Oui	Oui	Tout

LNA = Locked Nucleic Acid

Mfold = Un logiciel de prédiction de structure secondaire

SW = Smith-Waterman

FISH = Fluorescent In Situ Hybridization

EST = Expressed Sequence Tag

PCR = Polymerase Chain Reaction

De plus, il existe d'autres critères comme la position par rapport à l'extrémité 3' de la séquence de référence lorsqu'il s'agit d'étudier l'expression des gènes (la reverse transcription pouvant causer des artéfacts dus à des arrets prématurés de synthèse du premier brin d'ADNc). Il y a aussi des critères moins fréquemment intégrés dans les logiciels (car il n'a pas été démontré leur importance dans la sélection des sondes) comme la longueur optimale de la sonde calculée par le logiciel ou les motifs indésirables dans la sonde. Enfin, on trouve également le critère d'appariements multiples de différentes régions de la cible (Kreil et al., 2006).

D'un point de vue logiciel, trois catégories de programmes de conception de sondes pour biopuces à ADN sont distinguées:

Les logiciels graphiques stand-alone: ce sont les applications indépendantes qui fonctionnent à l'aide d'une interface graphique et qui sont téléchargeables et qui s'installent sur

une machine en local ou s'effectuera le calcul. Ce type d'application a l'avantage de fonctionner de manière indépendante mais requiert la plupart du temps des connaissances informatiques pour l'installation et l'utilisation suivant les plateformes informatiques cibles. (Picky, Yoda, HPD, ProbeMaker, OliCheck, GoArrays,…).

Les logiciels à interface Web ou à architecture Client/Serveur: ce sont les applications qui profitent des protocoles internet pour se connecter aux serveurs où elles sont installées. Il s'agit également de logiciels composés d'une partie cliente téléchargeable par l'utilisateur et qui se connecte à un serveur distant. L'avantage de ce genre d'architecture est la possibilité de l'utiliser sans connaissance particulière en informatique. Mais l'inconvénient reste la dépendance de la disponibilité des serveurs distants (ProbeMaker, OligoFaktory, ProbeWiz, Oligodb, Phylarray,…).

Les logiciels en ligne de commande: ce sont les applications qui demandent une connaissance plus pointue en informatique. Il s'agit en général de paquetage téléchargeable et qu'on peut installer ou bien décompresser dans un répertoire sur une machine en local. Ce genre d'applications dépend souvent d'autres logiciels qui doivent être installés au préalable comme le BLAST, Mfold ou de langages de programmation Perl, C++, C#, Java. L'inconvénient reste alors la difficulté à les faire fonctionner correctement (Oligoarray, OliCheck, Oligopicker, Promide, ROSO, ProbeSelect, OligoSelect, Primegens,…).

L'une ou l'autre des solutions a ses avantages et ses inconvénients. Cependant, le logiciel dépend fonctionnellement de l'étude pour laquelle il a été conçu (Adebiyi, 2007). C'est pourquoi il est important de bien choisir parmi les nombreux outils libres et payants celui ou ceux qui répondent au mieux à la problématique en question. Dans le premier cas, la plupart du temps il est impossible de modifier la base de référence pour la recherche de spécificité et l'utilisateur dépend souvent de la disponibilité du serveur pour effectuer des calculs. Dans le deuxième cas, ces logiciels sont souvent difficiles à configurer et à utiliser par un non-informaticien et dépendent bien entendu d'autres programmes externes.

1.4. Les puces à ADN en écologie microbienne

Dans le cadre de la connaissance de la diversité microbienne et de la compréhension du fonctionnement des écosystèmes, les puces à ADN représentent un outil de choix (Bar-or et al., 2007). Nous avons vu que les puces à ADN (microarrays, biochip, gene chip) représentent une plateforme d'hybridation à haut-débit permettant une identification rapide et simultanée de plusieurs centaines de milliers de gènes différents en une seule expérience (Partie I.6). Dans le domaine de l'écologie microbienne, ces biopuces peuvent donc être utilisées avec différentes

finalités en fonction du type de sondes et du type de cibles utilisées. Elles peuvent permettre ainsi d'estimer la part relative de chaque population d'organismes dans un environnement donné et de suivre l'évolution dynamique de ces populations au cours du temps comme par exemple lors d'un processus de bioremédiation. Le plus souvent ce sont les biopuces dites phylogénétiques qui sont employées. Celles-ci utilisent généralement les données de séquences d'ADNr 16S pour identifier les espèces composant une communauté microbienne naturelle.

De nombreux types de biopuces ADN sont utilisables en écologie microbienne (Cook and Sayler, 2003 ; Ehrenreich, 2006 ; Gentry et al., 2006 ; Loy and Bodrossy, 2006 ; Sessitsch et al., 2006 ; Wagner et al., 2007 ; Zhou and Thompson, 2002). Les biopuces dites fonctionnelles ou métaboliques peuvent permettre l'exploration des potentialités métaboliques d'un écosystème donné (He et al., 2007).

Dans le cadre de l'écologie microbienne portant sur la description de la biodiversité des écosystèmes complexes et sur leurs fonctionnements, nous présenterons principalement les deux types de biopuces pouvant contribuer à une meilleure connaissance de ces environnements: les biopuces fonctionnelles et les biopuces phylogénétiques.

1.4.1. Les biopuces fonctionnelles

Ce sont des biopuces appelées FGA (Functional Gene Array) ciblant des gènes fonctionnels qui sont utilisés pour l'étude des gènes impliqués dans les grands cycles biogéochimiques et dans les procédés de bioremédiation (Bodrossy et al., 2003 ; Rhee et al., 2004 ; Wu et al., 2004 ; Zhang et al., 2007). Les sondes utilisées peuvent être soit des produits PCR (500 à 1000 pb), soit des oligonucléotides (15 à 70 pb) (Sessitsch et al., 2006). Cependant, ce sont ces derniers qui sont généralement favorisés car les produits PCR, du fait de leur taille, peuvent engendrer des croisements aspécifiques (hybridations croisées). Les sondes oligonucléotidiques longues (~40- à 70-mers) sont préférées aux sondes courtes car, d'une part, elles sont plus sensibles et, d'autre part, leur spécificité, moins importante que celle des oligonucléotides courts, est compensée par le fait que les gènes fonctionnels présentent suffisamment de variabilité (Kostic et al., 2005). Ce point est cependant discutable en fonction de la complexité de l'écosystème et des cibles utilisées (produits PCR, ADNg, ARN). Les oligonucléotides de 50 mers permettent de différencier les cibles possédant en général moins de 88 % de similarité avec leurs sondes si les conditions d'hybridation sont suffisamment stringentes (T= 50° C ; %Formamide= 50%) (Rhee et al., 2004). La GeoChip composée de 24 243 oligonucléotides de 50 mers ciblant plus de 10 000 gènes impliqués dans les cycles du carbone, de l'azote, du soufre, du phosphore mais aussi dans la dégradation de contaminants organiques et

dans la réduction et la résistance aux métaux est la biopuce fonctionnelle la plus aboutie (He et al., 2007). Elle est le prolongement d'une biopuce de 1 662 sondes ciblant 2 042 gènes impliqués dans la biodégradation et dans la résistance aux métaux (Rhee, et al., 2004). La détermination des sondes avec le programme CommOligo est basée sur la recherche de séquences spécifiques des cibles ne montrant pas plus de 90% d'identité sans région de 20 bases consécutives identiques avec des gènes non cibles (Li et al., 2005). De plus, l'énergie libre de liaison des sondes avec des gènes non cible doit être ≥ 35 kcal/mol. Signalons que la séquence des sondes n'a pas été divulguée par les auteurs.

La majorité des études utilisant les FGA sont réalisées avec des cibles ADN de type produits d'amplification PCR permettant de déterminer si les gènes des organismes de l'environnement étudié sont présents. Les auteurs parlent ainsi de biopuces diagnostic permettant d'appréhender la structure des communautés microbiennes (Bodrossy et Sessitsch, 2004). Cependant, pour obtenir des informations sur les expressions des gènes d'intérêt, il est nécessaire d'utiliser des cibles ARN messagers (ARNm). Grâce à l'utilisation de telles molécules, Rhee et al., (2004) ont montré la présence et l'expression de gènes impliqués dans la dégradation du naphtalène. Il faut cependant noter que la plupart des études, utilisant les ARNm, sont réalisées sur des cultures pures, sur des systèmes simples ou sur des enrichissements en raison de la difficulté à extraire ce genre de molécules à partir d'environnements complexes (Gentry et al., 2006). Une biopuce diagnostic ciblant les gènes impliqués dans l'oxydation du méthane a récemment été hybridée avec des cibles provenant de l'extraction d'ARN (Bodrossy et al., 2006). La plupart des approches de détermination des sondes se base sur les séquences présentes dans les bases de données ce qui limite l'aspect exploratoire de ces biopuces. Il est cependant possible de déterminer des sondes exploratoires en considérant la dégénérescence du code génétique comme cela a été le cas pour définir une biopuce prototype ciblant les gènes de nodulation nodC pour la fixation d'azote (Bontemps et al., 2005). Le logiciel MetabolicDesign développé dans nos laboratoires permet une reconstruction *in silico* des voies métaboliques et la détermination de sondes spécifiques et de sondes exploratoires. Les biopuces fonctionnelles permettent donc de déterminer la présence, les niveaux d'expression de gènes connus voir d'identifier de nouveaux gènes.

1.4.2. Les biopuces phylogénétiques

Les biopuces phylogénétiques oligonucléotidiques ou « Phylogenetic Oligonucleotide Arrays » (POA) sont actuellement le type de biopuces le plus largement utilisé pour étudier les communautés bactériennes comme nous l'avons évoqué précedemment. Grâce aux sondes oligonucléotidiques, ciblant les ARNr 16S, les POAs permettent la discrimination des groupes

procaryotiques (Guschin et al., 1997; Small et al., 2001). Cependant, comme les séquences des ARNr restent très conservées, la détermination de sondes spécifiques de microorganismes peut se révéler être difficile (Zhou et Thompson, 2002). Ces dernières années, de nombreuses études utilisant les biopuces phylogénétiques oligonucléotidiques ont été réalisées pour explorer la diversité microbienne de différents environnements tels que les écosystèmes lacustres, les estuaires, les sols, les boues activées, les aérosols urbains (Brodie et al. 2007). La majorité de ces biopuces comportent des sondes ciblant différents niveaux taxonomiques selon l'approche décrite pour l'identification des entérocoques (Behr et al., 2000). Ces biopuces ont été utilisées pour cibler des groupes microbiens fonctionnels comme dans (Kyselková et al., 2008) par exemple pour *Actinomycetes Genera* ou dans (Bavykin et al., 2008) pour *Bacillus Anthracis*, des groupes taxonomiques ou pour appréhender l'entière diversité des environnements complexes.

Dans ce dernier cas, une biopuce de 7167 sondes uniques de 40 mers ciblant 359 espèces a été utilisée pour explorer des biopsies de colon (Palmer et al., 2006). Plus récemment, des changements de composition de la flore gingivale d'individus d'origines géographiques différentes ont été mis en évidence avec une biopuce de 9500 sondes de 25 mers (Huyghe et al., 2008). Une biopuce encore plus complexe ciblant un plus grand nombre d'espèces a été validée avec 62 358 sondes de 20 mers dirigée contre la région 1409-1491 en référence à *Escherichia coli* (DeSantis et al., 2005 ; Wilson et al., 2002). Une version haute densité de cette biopuce Affymetrix permet maintenant la détection simultanée de près de 9000 unités taxonomiques opérationnelles (Brodie et al., 2006). Cette PhyloChip est constituée de 506 944 sondes de 25 mers. Elle a permis de suivre la dynamique des populations bactériennes durant la réduction et la ré-oxydation de l'uranium. De même, la communauté des procaryotes de trois types d'environnements différents (un sol contaminé, un aquifère contaminé bio-stimulé et des échantillons d'air) a pu être étudié avec cette même biopuce (DeSantis et al., 2007). Cette dernière étude démontre que l'utilisation des biopuces permet de révéler une plus grande part de la diversité des procaryotes que l'approche clonage/séquençage. Par contre, les banques de clones ont permis quant à elles de mettre en évidence la présence de nouvelles séquences. Cette PhyloChip a également permis la comparaison des communautés bactériennes de l'air atmosphérique de deux villes américaines (Brodie et al., 2007). Près de 1800 types bactériens différents ont été détectés. Ces différentes études confirment donc que les biopuces sont particulièrement bien adaptées pour l'étude globale des différents groupes microbiens.

En conclusion, l'utilisation des biopuces pour l'étude de la biodiversité microbienne des écosystèmes peut présenter dans les deux cas (FGA et POA) une forte complexité en raison de la parenté des biomarqueurs utilisés (POA) et de la dégénérescence du code génétique (FGA).

Actuellement il n'existe aucune biopuce capable d'appréhender l'entière diversité des écosystèmes.

1.4.3. La dégénérescence des sondes oligonucléotidiques

Depuis la découverte des codons qui représentent le codage de l'information nucléique en information protéique en 1961 par Francis Crick et Sidney Brenner et la découverte du code génétique par Marshall Niremberg, Heinrich Mathaei et Severo Ochoa en 1966 les données génomiques ont augmenté de manière permanente et exponentielle grâce aux programmes de séquençages des génomes (Genome Online Database[1]) des êtres vivants. Par définition, l'ARN messager, transcrit à partir de l'ADN, a une fonction principale, celle de spécifier une protéine à synthétiser. Une grande partie des autres ARN cellulaires participent aussi à cette tâche, mais en tant que machines à décoder plutôt que de porteurs du code. Le code à déchiffrer comporte quatre lettres, correspondant aux nucléotides (A, G, C, U ≈ T), alors que sa traduction en compte 20, les acides aminés qui composent toutes les protéines. Le nombre de nucléotides nécessaires pour spécifier un acide aminé est donc de trois. Avec trois nucléotides, le nombre de combinaisons est de 64, ce qui permet d'incorporer une ponctuation, nécessaire à la bonne lecture du code, et implique une redondance, avec plusieurs triplets de nucléotides spécifiant le même acide aminé. On parle alors de dégénérescence du code génétique. Le Tableau 6 donne les codons standards des 20 acides aminés avec leurs codes SLC (Single-letter Database Code) à une lettre. La prise en compte de la dégénérescence du code génétique devrait permettre une évolution significative des biopuces fonctionnelles qui ne se servaient jusqu'à présent qu'à des données de séquences géniques. Comme nous le verrons ultérieurement nous proposerons une approche alternative de design de sondes pour biopuces fonctionnelles en utilisant la dégénérescence du code génétique pour cibler l'ensemble des gènes codant pour une protéine donnée (algorithme HiSpOD).

Tableau 6: Les 20 acides aminés et les codons STOP et leurs codes ADN correspondants.

Acide Aminé	Code SLC	Codons ADN
Isoleucine	I	ATT, ATC, ATA
Leucine	L	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	GTT, GTC, GTA, GTG
Phénylalanine	F	TTT, TTC
Méthionine	M	ATG
Cystéine	C	TGT, TGC
Alanine	A	GCT, GCC, GCA, GCG

Glycine	G	GGT, GGC, GGA, GGG
Proline	P	CCT, CCC, CCA, CCG
Thréonine	T	ACT, ACC, ACA, ACG
Serine	S	TCT, TCC, TCA, TCG, AGT, AGC
Tyrosine	Y	TAT, TAC
Tryptophane	W	TGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAT, AAC
Histidine	H	CAT, CAC
Acide Glutamique	E	GAA, GAG
Acide Aspartique	D	GAT, GAC
Lysine	K	AAA, AAG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Codons STOP	Stop *	TAA, TAG, TGA

Dans le cas des biopuces taxonomiques, certains logiciels [DEODAS utilisant CODEHOP (Rose et al., 1998), BussuB (López-Campos et al., 2007), Phylarray (Militon et al., 2007) et PRIMROSE (Ashelford et al., 2002)] utilisent le code dégénéré IUB (Tableau 7) pour concevoir des sondes ADN dégénérées qui ont l'avantage de ne plus être spécifiques à une séquence en particulier mais à un groupe de séquences souvent ayant un lien phylogénétique ou fonctionnel. Cette approche peut jouer un rôle exploratoire mais rajoute une complexité supplémentaire aux algorithmes de recherche et de sélection de sondes optimales.

Tableau 7: Codes IUB.

Code	Définition	Mnémonique
A	Adénine	**A**
C	Cytosine	**C**
G	Guanine	**G**
T	Thymine	**T**
R	A/G	pu**R**ine
Y	C/T	p**Y**rimidine
K	G/T	**K**eto
M	A/C	a**M**ino
S	G/C	**S**trong
W	A/T	**W**eak
B	C/G/T	Not A
D	A/G/T	Not C
H	A/C/T	Not G
V	A/C/G	Not T
N	A/G/C/T	a**N**y

Incontestablement, et dans ce contexte de croissance exponentielle des bases de données de séquences, il est obligatoire de résoudre le problème de performance des algorithmes de conception de sondes autrement, c'est-à-dire en s'aidant des architectures parallèles et distribuées à la fois pour distribuer le calcul et pour stocker des données de plus en plus importantes. Une partie des algorithmes et applications de conception de sondes ADN ont un temps d'exécution relativement long [OligoArray 1.0 (Rouillard et al., 2002), CommOligo (Li et al., 2005) et ArrayDesigner 3.0] (Adebiyi, 2007) ou ArrayOligoSel 3.5 (Leiske et al., 2006). En effet, OligoArray met sur une CPU de 1.2 GHz environ 12 heures pour calculer jusqu'à 3 sondes de 45 mers pour un gène donné par génome bactérien. CommOligo met quant à lui, environ 18 heures pour sélectionner les sondes du génome de *M.Maripaludis* sur une CPU de 2.5 GHz. ArrayDesigner met 84 heures pour les 5800 gènes de *S. Cerevisiae*. ArrayOligoSel met environ 10 heures pour sélectionner des sondes pour le génome de *B. Cenocepacia*. Enfin, Phylarray met 8 jours environ pour le genre *Bacillus* sur une machine SMP de 2.18 GHz en utilisant 14 cœurs.

Des efforts considérables ont été initiés par les scientifiques pour proposer des outils permettant de concevoir des oligonucléotides répondant à des critères bien précis qui assurent leur qualité (en termes de spécificité et de sensibilité). Malgré ces travaux et le nombre d'applications disponibles, les outils proposés – souvent difficiles à utiliser et/ou compliqués à installer (Adebiyi, 2007) – manquent de qualité en termes de conception informatique et de génie logiciel. La plupart ont des temps d'exécution très longs et favorise la spécificité sans se focaliser sur la performance de calcul. D'une part, avoir recours aux architectures informatiques parallèles et distribuées ne peut qu'être bénéfique devant les défis de performance et de stockage de données soulevés. D'autre part, le génie logiciel peut apporter un niveau d'abstraction supplémentaire et contribuer à la simplification des approches de sélections de sondes en automatisant certains composants logiciels. La partie suivante présente les avancées en termes d'ingénierie dirigée par les modèles pouvant contribuer à une meilleure conception des logiciels de conception de sonde pour biopuces à ADN.

2. L'ingénierie dirigée par les modèles (IDM)

L'ingénierie dirigée par les modèles – qu'on notera IDM par la suite (de l'anglais Model Driven Engineering (MDE)) – est une famille d'approches de modélisation et de description à la fois des parties indépendantes des plates-formes spécifiques, au travers de PIM (Platform Independant Models), et des parties liées aux plates-formes, grâce à des PSM (Platform Specific Model), en s'appuyant sur le standard UML (Unified Modeling language). D'après 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. De plus, elle se veut productive en automatisant la prise en charge des outils relatifs à la validation des modèles, les transformations et les générations de code […]. L'IDM cible une production logicielle bien fondée […]* ». C'est une intégration, un prolongement et un renforcement d'approches déjà connues. On retrouve donc les notions fondamentales de langage, de méta-modèle et de transformations qui sont les définitions essentielles du projet. Ce projet a été initié par l'OMG (Object Management Group) [http://www.omg.org/] qui met en avant une architecture basée sur le MOF (Meta-Object Facility) s'appuyant d'une part sur une collection de méta-modèles dont UML n'est qu'une approche parmi d'autres, et d'autre part sur une technologie de transformation de modèles MOF/QVT (Query / View / Transformation) qui fera l'objet d'un standard (Favre et al., 2006). L'approche MDA (Model Driven Architecture) vise à proposer une méthode dirigée par les modèles pour le développement et la maintenance des systèmes à prépondérance logicielle. L'IDM se veut une approche unificatrice pour tout type de système et s'inscrit dans une évolution logique des standards préétablis. En effet, la notion d'objet n'est plus le cœur du développement logiciel, mais d'autres telles que le modèle, le méta-modèle, le langage et la transformation font leurs preuves (Greenfield et al., 2004).

2.1. Définitions

2.1.1. MDA (Model Driven Architecture)

Suite aux évolutions permanentes subies par les architectures middleware, les langages de programmation et les méthodes de génie logiciel, en novembre 2000, l'OMG rend public l'architecture MDA basée sur une utilisation intensive de l'UML (Unified Modeling Language) pour séparer les spécifications fonctionnelles d'un système des spécifications de son implémentation. L'OMG place la modélisation au centre du processus de développement (Figure 11) en découplant la logique métier, la logique applicative et le code. Dans MDA, la partie

centrale est de construire un PIM représentant le modèle métier en utilisant les technologies UML, MOF et CWM (Common Warehouse Meta-Model). La deuxième étape consiste à transformer le PIM en PSM qui représente un modèle comprenant des parties techniques propres à la plateforme cible basée sur les technologies telles que EJB (Entreprise Java Beans), CORBA (Common Object Request Broker Architecture), .NET, XMI (XML Metadata Interchange) ou encore les web services. La dernière étape consiste à générer du code applicatif à partir du PSM pour une plate-forme spécifique au domaine d'application du logiciel.

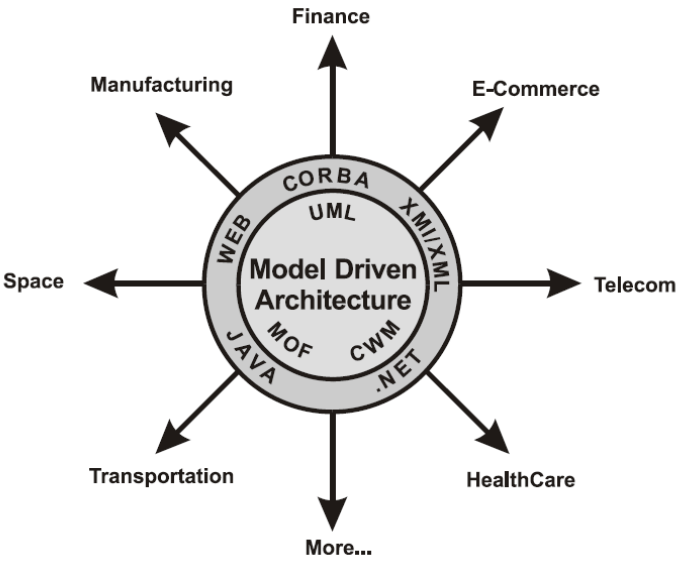


Figure 11: L'architecture MDA.

Plusieurs outils proposent effectivement la transformation suivant les standards MDA des PIM en PSM. Aujourd'hui, il est concevable de s'appuyer sur cette architecture pour concevoir des applications WEB, des systèmes embarqués ou encore des systèmes d'information entiers. Parmi les outils proposés on trouve : OptimalJ (plate-forme J2EE) de CompuWare[1] qui est proposée dorénavant sous forme plus exhaustive au niveau des langages de programmation cible avec ACCELEO[2] (Figure 12), ATLAS Transformation Language[3], MODEL-IN-ACTION de Mia Software[4], ANDROMDA[5], OpenMDX[6] [plates-formes J2EE et .NET], MDE Eclipse[7], Blueprint ME de @PORTUNITY[8] ou encore Visual Studio (VS) de Microsoft avec son Entity Designer intégré à l'IDE de VS pour le développement d'applications .NET Entreprise.

[1] http://www.compuware.fr
[2] http://www.acceleo.org
[3] http://www.sciences.univ-nantes.fr/lina/atl/
[4] http://www.mia-software.com
[5] http://www.andromda.org
[6] http://www.openmdx.org
[7] http://sourceforge.net/projects/mde
[8] http://www.atportunity.com

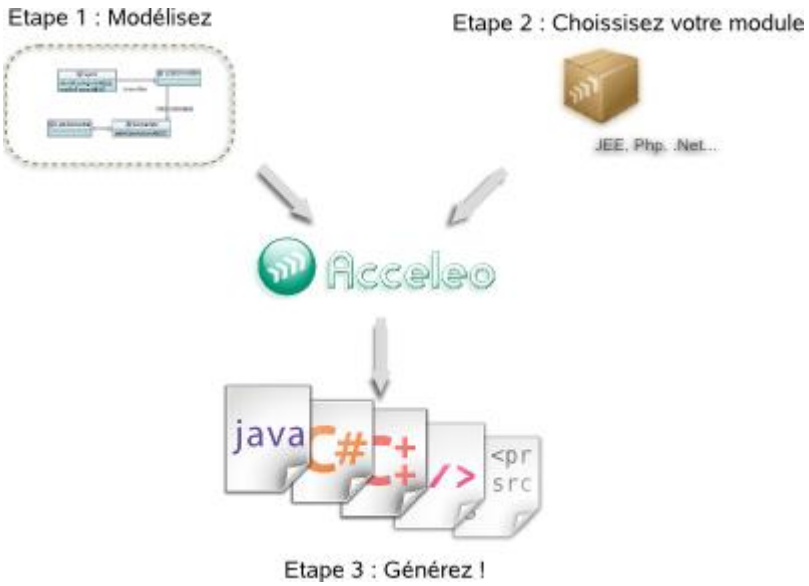


Figure 12: Exemple d'implémentation de l'architecture MDA. Extrait du site internet d'ACCELEO.

(Source : http://www.acceleo.org)

Bien que l'architecture MDA soit suffisamment bien définie dans le cadre du génie logiciel pour répondre à la problématique d'abstraction et aux notions fondamentales de *modèle*, de *méta-modèle*, de *transformation* et de *plate-forme*, il est difficile de définir de manière précise et sans ambigüité la notion de plate-forme par exemple qui tend à disparaitre au profit des notions de modèle et de méta-modèle (Favre et al., 2006). La relation entre ces notions est loin d'être définitivement caractérisée et l'ingénierie dirigée par les modèles s'inscrit dans cette logique évolutive pour proposer des concepts unificateurs basés sur les modèles afin de généraliser et d'intégrer tout type de modèle.

2.1.2. Modèle, Méta-modèle et Système

Dans les années 80, le principe fondamental sur lequel ont été basées toutes les approches technologiques était « *Everything is an object* ». Depuis, le principe est transformé pour que le système ne soit plus un objet du point de vue abstraction, mais plutôt un modèle avec le nouveau principe « *Everything is a model* » (Bézivin, 2004). C'est la preuve que les notions ont largement évolué pour permettre de modéliser non plus les composants logiciels uniquement mais aussi les différentes étapes de la création d'un outil.

Paradoxalement, il n'existe pas de définition universelle pour le terme « *Modèle* » d'après Jean–Marie Favre. Néanmoins, plusieurs définitions apparues dans la littérature peuvent donner un aperçu de ce qu'est un modèle. En effet, un modèle peut désigner « *l'abstraction d'un système construit dans une intention particulière pour un rôle bien défini par rapport à un autre système* » ou « *un ensemble de faits sur un système considéré* » ou « *une description (d'une partie) d'un système dans un langage bien*

défini » ou encore « *un modèle doit répondre aux questions à la place du système réel* ». Pour définir les relations entre modèles (représentations abstraites des systèmes), il est nécessaire d'utiliser les notions d'inclusion, d'appartenance et de représentation ou d'interprétation.

Un méta-modèle quant à lui est défini comme étant « *un modèle qui définit le langage pour exprimer un modèle* » ou aussi « *un modèle de spécification d'une classe de systèmes où chaque système de la classe est lui-même un modèle valide exprimé dans un certain langage de modélisation* ». Les notions de langage et de méta-modèle sont souvent confondues mais concrètement, un méta-modèle est finalement un moyen de définir un langage.

Un système du point de vue de l'IDM est tout élément du discours. C'est donc ce qui fait l'objet de la modélisation réalisée par un modèle. Mais tout système peut jouer le rôle d'un modèle par rapport à un autre système. En conclusion, l'IDM est basée en partie sur ces trois notions fondamentales tirées de l'architecture MDA de l'OMG et sont parfois difficiles à assimiler (Favre et al., 2006).

2.1.3. Transformation de modèles

Pour rejoindre les définitions précédentes, un modèle est vu comme une interprétation du monde réel. L'une des notions les plus importantes introduite par l'OMG pour définir la fonction qui permet de passer d'un modèle à un autre est la transformation de modèles. Dans le cadre de l'IDM, on parle de modèle de transformation pour désigner un programme, exécutable ou non, permettant de réaliser une conversion du système source en un système cible. L'objectif est de proposer une automatisation de telles fonctions dans l'approche de l'ingénierie dirigée par les modèles. Dans un processus de transformation basée sur l'IDM, les règles de transformations sont décrites dans les spécifications (Figure 13).

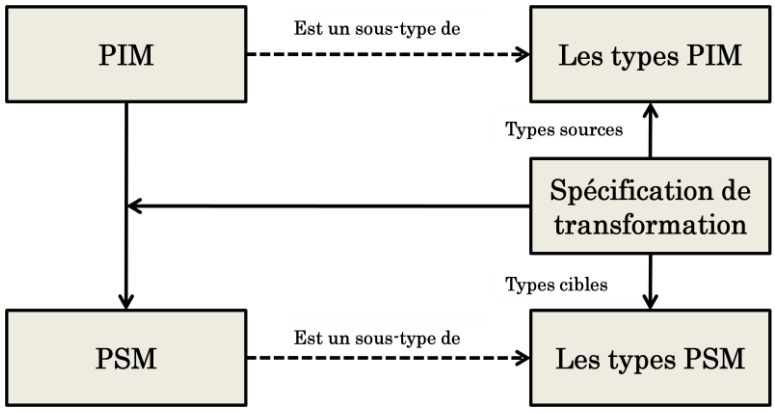


Figure 13: Approche de transformation d'un modèle basée sur la liaison entre les types du modèle source (PIM) et les types du modèle cible spécifié par l'OMG.

En cas de besoin, le modèle cible est créé en se basant sur le modèle source. La complexité de cette tâche a même été simplifiée en ajoutant un nouveau concept : le méga-modèle qui vise à modéliser l'évolution des logiciels à travers les transformations (Favre et Nguyen, 2005) dans le cadre de la MDE. D'après (Sendall et Kosaczynski, 2003), un langage de transformation de modèle doit:

- Etre exécutable,
- Etre implémenté efficacement,
- Etre totalement expressif et sans ambigüité pour les transformations de modèles existants ou pour la création de nouveaux modèles,
- Faciliter la productivité du développeur avec précision, concision et clarté,
- Fournir un moyen de combiner des transformations pour former des composites, une sélection conditionnelle et une répétition des transformations,
- Fournir un moyen de définir les conditions de l'exécution des transformations.

Certaines transformations ne sont pas automatisables et dans ce cas là elles ne sont pas intéressantes pour une approche par IDM, comme par exemple la correction de bogues dans des programmes informatiques. A titre d'exemples, on peut citer SmartQVT (http://smartqvt.elibel.tm.fr/) qui est une implémentation JAVA du langage QVT des spécifications MOF 2.0, le projet M2M (Model-2-Model) de Eclipse qui vise à proposer une infrastructure composée de 3 moteurs de transformations [ATL (Atlas Transformation Language), QVT procédural et QVT déclaratif (tous deux basés sur MOF 2.0)], ou encore le projet Kermata de Triskell (www.kermata.org) basé sur une extension de la MOF qui s'inspire du MTL (Model Transformation Language) et du OCL (Object Constraint language). On peut également citer XAL (Xion Action Language) (Heitz et al., 2007) basé sur les diagrammes d'activités UML et XMLTL (XML Transformation Language) qui permet de transformer un document XML en une sortie XML. Plusieurs implémentations sont disponibles [XSLT (eXtensible Stylesheet Language Transformation), WQuery, STX, XML Script, FXT (Functional XML Transformation), XDuce, XStream…). Enfin citons VIATRA (Visual Automated Model Transformation) un environnement de vérification et de validation basé sur la transformation de modèles implémenté en UML et permettant d'accroitre la qualité des systèmes en vérifiant de manière automatique la consistance, la complétude et la dépendance des spécifications d'un système.

2.1.4. Méta-modélisation et méta-programmation

La méta-modélisation dans une approche IDM est la définition d'un méta-modèle qui décrit une famille de modèles. L'OMG a standardisé l'approche à l'aide de la MOF pour fournir un outil générique permettant de manipuler ces concepts. L'approche comprend 4 niveaux d'abstraction (Figure 14):

Le niveau M0: il définit le système réel en lui-même.

Le niveau M1: il définit le modèle du système dans un certain langage.

Le niveau M2: il définit le méta-modèle qui décrit ce langage.

Le niveau M3: il définit le méta-méta-modèle qui décrit le méta-modèle (couche auto descriptive).

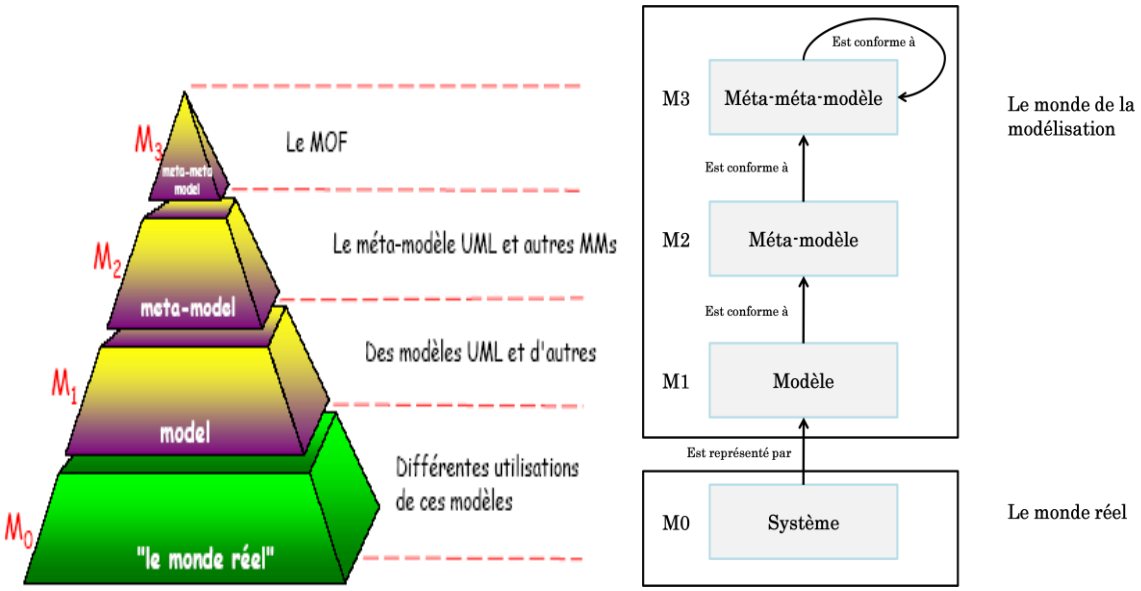


Figure 14: Les niveaux d'abstraction selon la MOF du MDA.

A l'aide de cette approche à 4 couches, tout modèle peut être décrit par une syntaxe, une sémantique, et des règles selon la MOF, même les modèles de transformation.

De façon analogue à la méta-modélisation, on peut définir la méta-programmation comme étant la méta-modélisation de programmes informatiques. Cependant, il ne faut pas confondre IDM et méta-programmation même si les deux partagent quelques concepts: les programmes sont des modèles (ex. classe JAVA) sur lesquels on peut effectuer des opérations de transformation (ex. méthodes) de manière automatique (ex. un calcul particulier) et ayant plusieurs représentations possibles (ex. C++, C#...). Cette analogie est simpliste (Tableau 8) mais permet de formaliser les concepts de l'IDM de manière concrète (Batory, 2006). Si on prend

l'exemple d'un MAKEFILE, il est évident qu'il s'agit d'un méta-modèle (méta-programme, par analogie à l'IDM) permettant de générer un modèle (respectivement un programme). L'idée de base dans l'ingénierie dirigée par les modèles dans ce cadre est la capacité à créer des langages personnalisés pour un domaine d'application particulier et de définir leurs spécifications. Les méta-modèles sont alors la pièce maîtresse de la définition même des DSL (Domain Specific Language) tirés de l'IDM. On peut étendre cette analogie aux bases de données pour se rendre compte que les notions de l'IDM ne sont rien d'autres que des représentations de concepts familiers.

Tableau 8: Correspondances entre les terminologies MDE, les bases de données et les langages Orienté Objet d'après (Batory, 2006).

MDE	Bases de données	Programmation
Méta-méta-modèle	Schéma d'information	Méta-classe
Méta-modèle	Schéma	Classe
Modèle	Base de données	Objet
Transformation	Transaction	Méthode

Par analogie cette fois à la programmation, la synthèse d'une application à partir de son modèle par génération de code est un processus à 2 étapes qui commence par une transformation du modèle en langage de programmation de haut niveau (C, C++, JAVA,…) et se termine par la compilation de ce programme en binaires exécutables qui réalise l'application par l'utilisation d'un compilateur support du langage ciblé lors de la première étape (Favre et al., 2006). La génération de code peut être comprise comme un ensemble de règles de transformation de modèles et peut être optimisée pour une plate-forme donnée. Les patrons de conception (les design patterns) peuvent alors implémenter ces règles.

Enfin, dans une démarche MDE, la génération de code semble adaptée car elle offre une flexibilité face à l'évolution permanente des standards des langages de modélisation et face à la diversité des plates-formes et des systèmes. Seule nuance, reste la gestion des corrections (débogage) qui n'est pas une tâche triviale et se fait généralement au niveau du code généré et non pas au niveau du modèle. Une solution envisageable est celle d'un interpréteur de modèle qui puisse anticiper les erreurs à partir du modèle.

Plusieurs outils permettant la génération de code automatisée sont disponibles: GreenBox[1] pour les plates-formes JAVA, ALTOVA[2] basé sur des modèles décrits en XML ou en UML, Codus[3] permettant la génération de code pour la mise à jour des bases de données et bien d'autres outils[4].

2.2. Processus d'ingénierie dirigée par les modèles

Définir une méthodologie intégrée d'ingénierie est loin d'être une tâche facile. En effet, une telle démarche impliquerait un travail considérable de la part de spécialistes dans tous les domaines relatifs à l'approche: les plates-formes, le test, la vérification, la validation… (Fondement et Silaghi, 2004). L'idée de définir un tel processus a l'avantage de définir clairement chaque étape du développement d'un système en forçant les ingénieurs (développeurs) à suivre une méthodologie unique qui leur permet de concevoir des modèles de la même manière qu'ils développent du code. Ce processus doit spécifier la séquence de modèles à développer et la façon dont il est possible de dériver un modèle d'un autre. Cela permet aux développeurs de connaitre à tout moment du cycle de développement ce qui reste à faire et comment. De plus, il est important de noter que l'un des avantages d'une démarche IDM est la possibilité d'absorber les modifications des changements quelque soit leur niveau d'abstraction: élevé au niveau du modèle (ex. nouveaux besoins du système modélisé) ou bas au niveau du système (ex. migration vers une nouvelle plate-forme). D'où l'intérêt d'avoir un langage de modélisation adaptatif en proposant un module de raffinement pour mettre à jour les parties changeantes. Un besoin essentiel dans une telle approche est la standardisation des techniques d'ingénieries et la définition d'une démarche unifiée pour le développement de système (Alanen et al., 2003). L'OMG a proposé une démarche standardisée appelée SPEM (Software Process Engineering Meta-Model) en 2002 avec la première version 1.0. Aujourd'hui, la version 2.0 s'intitule Software & **System** Process Engineering Meta-Model. Ceci traduit d'une manière claire la volonté d'une orientation vers un domaine de l'ingénierie dirigée par les modèles plus générique et plus fédérateur qui ne concernera plus uniquement le génie logiciel mais qui s'étendra à d'autres systèmes.

2.2.1. Principe de l'IDM

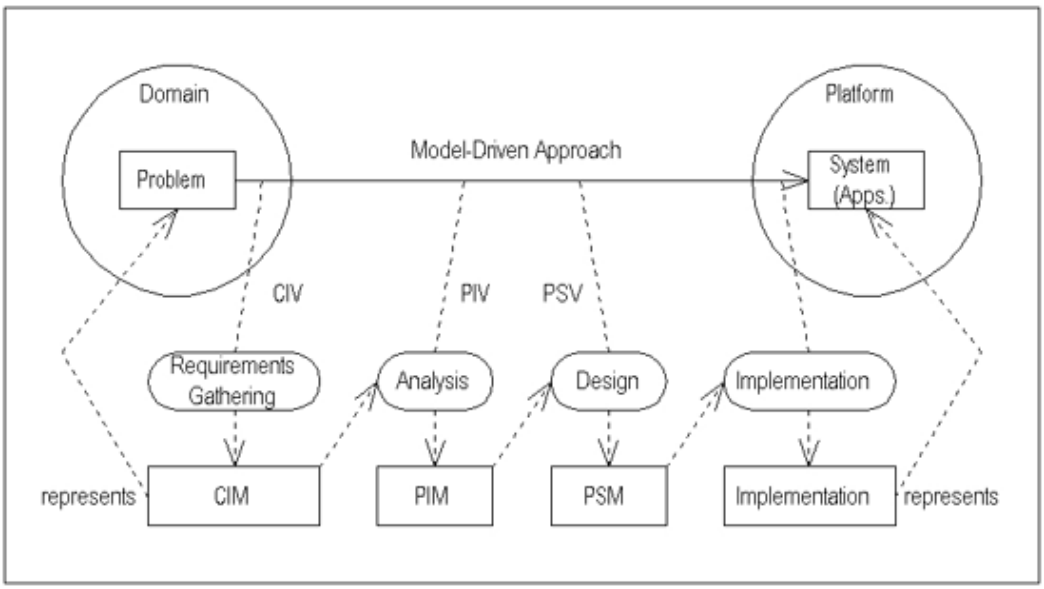
Fondement et Silaghi, (2004) proposent une définition générale du processus de l'IDM. Ils la définissent comme étant une méthode qui permet de modéliser un problème ainsi que sa solution à différents niveaux d'abstraction et qui fournit une plate-forme capable de dire quel

[1] http://greenbox.dev.java.net
[2] http://www.altova.com
[3] http://sharptoolbox.com/redirect/tools/codus
[4] http://sharptoolbox.com

modèle utiliser à quel moment. Un tel processus est défini à l'aide de propriétés que les spécialistes de méthodologies doivent fournir. Ces propriétés qui sont censées définir complètement l'IDM ne sont malheureusement pas claires, bien que les techniques qui permettent de l'utiliser aient été largement étudiées.

L'IDM est une alternative de plus en plus nécessaire aux développeurs qui voient le nombre de lignes de codes de leurs programmes augmenter sans arrêt et la complexité des applications informatiques croître sans cesse. Elle peut permettre aux développeurs de simplifier leurs programmes en définissant clairement les méthodologies, en développant des systèmes à n'importe quel niveau d'abstraction et en organisant les activités de validations et de tests (Azmoodeh et al., 2005). Ces différentes tâches demandées à l'IDM sont résolues indépendamment les unes des autres par des solutions informatiques opérationnelles. Le plus difficile est d'exprimer toutes ces contraintes à l'aide de modèles qui soient compréhensibles à la fois par l'homme et par la machine. Les modèles peuvent être utilisés à n'importe quel niveau de l'abstraction du système et peuvent faire l'objet de la modélisation d'une partie du système. Dans le cadre de l'IDM, et puisqu'on demande qu'un modèle soit transformé par une machine, le processus de développement est devenu un processus itératif qui transforme un modèle abstrait en modèle plus concret et qui génère et déploie le code complet de façon automatisée (Jouault et Kurtev, 2007).

Plusieurs outils d'ingénierie ont été développés durant les dernières années, tous sont spécifiques à un domaine ou à une plate-forme. Le challenge est donc de proposer un standard auquel tous ces outils doivent répondre pour que les systèmes soient pilotés par ces outils d'ingénieries tout au long de leur cycle de vie (la Figure 15: Cycle de vie du développement des systèmes (Alhir, 2003). présente l'approche de conception d'un système avec toutes les phases de développement).



CIM=Computation Independent Model

CIV=Computation Independent Viewpoint

PIV=Platform Independent Viewpoint

PSM=Platform Specific Viewpoint

Figure 15: Cycle de vie du développement des systèmes (Alhir, 2003).

L'objectif est donc de préciser quel langage sera utilisé pour exprimer les modèles, comment effectuer des transformations de modèles, comment échanger des modèles, comment enregistrer et faire évoluer les modèles et récemment comment générer le code. En effet, tant qu'il existe des systèmes où l'IDM ne peut pas être appliqué, des problèmes liés au processus persisteront.

2.2.2. Mise en œuvre d'une démarche IDM

Un processus d'ingénierie dirigée par les modèles commence par une phase dans laquelle le système est entièrement décrit par un modèle à un niveau d'abstraction très élevé en ignorant les aspects d'implémentation ou de mise en œuvre concrète de la technologie qui sera utilisé (Bézivin, 2005). On parle alors de modèle indépendant de l'informatique (Computation Independant Model CIM). C'est un moyen de saisir les besoins du système sans se préoccuper de leurs implémentations en terme technique. Les diagrammes les plus adaptés pour une telle abstraction sont le diagramme de cas d'utilisation et les diagrammes orientés caractéristique (Feature-Oriented Diagram). Ensuite, une étape de raffinement est nécessaire pour rendre le modèle spécifique à une plate-forme en particulier. Le système est alors exprimé d'une autre manière en utilisant des modèles plus précis tels que le diagramme de classe UML et le diagramme d'état pour montrer son comportement réel. D'autres aspects et informations peuvent être également ajoutés pour intégrer les problèmes de mise en œuvre ou de production

concernant les middlewares avant la génération de code. Un modèle fini est appelé ainsi un modèle spécifique à une plateforme (PSM). Les informations relatives à la gestion de la qualité peuvent aussi être intégrées au modèle ainsi que la phase de vérification, de validation et de test.

Il est également essentiel de souligner l'importance de la transformation de modèle dans une démarche IDM. En effet, ceci garantirait une interopérabilité – *la capacité de deux ou plusieurs systèmes ou composants d'échanger de l'information et d'utiliser l'information échangée* – maximale entre différents langages et plates-formes. Le standard QVT proposé par l'OMG a l'avantage d'être compatible avec les autres standards OMG: UML, OCL et MOF. Mais la course extraordinaire menée par les compagnies et les instituts de recherche pour développer d'autres approches et langage de transformation de modèles a permis l'apparition de plusieurs langages séparés qui ont montré leur efficacité dans certains cas particuliers (Jouault et Kurtev, 2007). Dans (Mens et Gorp, 2006 ; Mens et al., 2006), une taxonomie décrivant une méthode permettant d'aider à choisir le meilleur langage de transformation dans un contexte donné a été présentée.

Enfin, il faut noter que SysML (System Modeling Language), une version étendue du langage UML pour l'ingénierie des systèmes, a été développé par l'OMG et permet de modéliser dans une optique IDM différents types de systèmes tels que les systèmes industriels. SysML est un moyen de regrouper dans un modèle commun à tous les corps de métiers, les spécifications, les contraintes, et les paramètres de l'ensemble du système. C'est un profil d'UML 2.0. Il permet en effet, le partage des spécifications des systèmes entre tous les corps métiers et documente leur savoir-faire. Une description de son utilisation est présentée dans (Willard, 2007). SysML permet de spécifier les systèmes, analyser la structure et le fonctionnement des systèmes, décrire les systèmes et concevoir des systèmes composés de sous systèmes et vérifier et valider la faisabilité d'un système avant sa réalisation.

2.2.3. Les avancées récentes en matière d'IDM

Depuis quelques années, l'ingénierie dirigée par les modèles intéresse de plus en plus d'entreprises et de structures académiques et industrielles. Son importance réside dans le fait qu'une telle approche puisse s'intégrer tout naturellement dans un processus de développement à base de modèles s'assurant, à chaque niveau de modélisation, que les modèles obtenus et réutilisés ont les qualités requises. Cette démarche dirigée par les modèles met le modèle au centre des préoccupations des analystes/concepteurs. L'élaboration des modèles devient donc centrale et le choix du formalisme revêt une importance capitale. Cependant, en débutant le processus avec UML, les modèles pourront être plus facilement accessibles. En revanche, en se basant dès

le départ du processus sur un langage de modélisation spécifique au domaine, le processus restera donc l'affaire des experts du domaine.

Les tendances actuelles d'après Fabrice Kordon (Kordon et al., 2008) s'orientent vers des propositions complémentaires à l'approche IDM. Il propose de l'ingénierie dirigée par les vérifications (VDE) qui permet de décrire comment modéliser un système dans le but de l'analyser en introduisant des propriétés aux modèles: les propriétés structurelles (connexions et interdépendances), les propriétés qualitatives (comportement du système) et les propriétés quantitatives (performance du système). Par ailleurs, dans (Rougemaille et al., 2009) les auteurs soulignent l'intérêt de SPEM (Software and System Process Engineering Meta-model) dans la conception de processus adaptatif de méthodologie. En effet la version 2.0 permet d'améliorer le contenu des méthodologies ainsi que la réutilisabilité des processus en séparant l'aspect dynamique de l'aspect structurel de la méthodologie employée. Les méthodologies ADELFE (Atelier de Développement de Logiciels à Fonctionnalité Emergente) et PASSI (a Process for Agent Societies Specification and Implementation) ont été retenues comme exemples concrets de méthodologies basées sur la décomposition en plusieurs sous-parties.

De plus, dans (Abdoul et al., 2008) un langage de modélisation appelé AADL (Architecture Analysis & Design Language) a été présenté et permet de couvrir en plus l'aspect exécution du système dans la transformation de modèle. Ce langage est adapté pour la modélisation des systèmes embarqués d'après les auteurs. Un des outils implémentant ce langage est Kermeta (www.kermeta.org/).

Les applications utilisant l'IDM comme base et démarche de conception sont d'abord les logiciels. Mais ces dernières années on remarque de plus en plus de systèmes de toute nature prendre du terrain. On trouve, par exemple, les systèmes embarqués à hautes-performance (Yu et al., 2008), les systèmes multimédia (Abdallah et al., 2009), les systèmes polysynchrones (Potop-Butucaru et al., 2009), les systèmes de surveillance maritimes (Monperrus et al., 2009) et les systèmes biologiques (Roux-Rouquié et Schuch Da Rosa, 2006) etc. Cette évolution s'inscrit complètement dans la perspective visée par l'IDM. En effet, l'objectif de cette démarche est de pouvoir modéliser tout type de système en utilisant un « standard » général capable de fédérer tout modèle. L'avenir de l'IDM dépendra des évolutions en terme de modélisation des processus métiers dans un cadre plus général et les solutions qui seront apportées en terme de modèle spécifique à des domaines en particulier et la capacité à rendre la transformation de ces modèles le plus simple possible.

Obeo[1], Crescendo[2], Atos[3], CS[4] et IBM[5] sont des acteurs majeurs dans l'évolution des technologies de la conception des systèmes d'information. Eclipse avec son Framework EMF est l'un des précurseurs dans le développement des plates-formes orientées IDM avec notamment Acceleo, EMF compare ou ATL. On peut également citer Topcased[6], un projet d'AeroSpace Valley basé sur la plate-forme d'Eclipse et qui permet la modélisation des systèmes critiques. Toutes ces sociétés ont pour objectif de proposer des solutions d'avenir pour d'intégration et l'automatisation du développement logiciel en se basant sur des approches par les modèles.

Enfin, il faut noter que dans la recherche permanente de système fiable et performant, le développement des logiciels s'oriente de plus en plus vers une implémentation distribuée et parallèle. Les exemples fars sont ceux d'IBM et d'Oracle. Nous présenterons dans la suite un état de l'art sur les architectures informatiques à haute performance.

[1] http://www.obeo.fr
[2] http://www.crescendo-technologies.com
[3] http://www.fr.atosorigin.com
[4] http://www.c-s.fr
[5] http://www.ibm.fr
[6] http://www.topcased.org

3. Le calcul parallèle et distribué

3.1. Introduction

Dans le chapitre I, une introduction au calcul haute-performance a été présentée. L'intérêt dans la bioinformatique est de proposer des applications capables de s'exécuter en un temps raisonnable (Page et McInerney, 2005) sachant que la quantité de données produite par la biologie moléculaire et disponible dans les banques de données de séquences est de plus en plus importante. Une des solutions adoptées depuis des années est de faire fonctionner les algorithmes bioinformatiques consommateurs de temps de calcul sur des architectures dites parallèles. Par définition, le calcul parallèle est l'utilisation simultanée de plusieurs ressources informatiques pour résoudre un problème. On introduit le calcul parallèle principalement pour: (1) gagner du temps, (2) résoudre des problèmes complexes et (3) faire de la concurrence (plusieurs tâches simultanément). Les temps de calculs de certains algorithmes bioinformatiques appliquées à une quantité de données importantes peuvent rapidement dépasser la capacité d'une seule machine (Zomaya, 2006). Dans le cadre de la conception d'oligonucléotides pour les biopuces à ADN, la recherche de sondes nécessite des ressources de calculs croissantes (Zhu et al., 2006 ; Simmler et al., 2003) et les architectures parallèles représentent un outil de choix pour répondre à ces besoins. Sachant que la plupart des applications de conceptions de sondes font appel souvent à des programmes de comparaison de séquences (voir section II.2) tels que BLAST, la parallélisation des ces programmes peut faire gagner beaucoup de temps (Oroguchi et al., 2005). Dans la suite, après avoir introduit les architectures informatiques, une partie sera consacrée aux architectures multiprocesseurs et aux fermes de calcul (clusters) avant de terminer par l'architecture de grille de calcul.

3.2. Les architectures parallèles : vue d'ensemble

3.2.1. La machine de Von-Neumann

Après avoir longtemps considéré les programmes informatiques comme une suite séquentielle d'instructions pouvant être exécutées sur un ordinateur appelé calculateur universel par Von-Neumann (voir Figure 16: Machine de Von-Neumann. Source Wikipedia traduction de l'anglais.), l'idée de parallélisation est apparue au début des années 50 pour proposer une alternative à la limite de la performance des processeurs qui reste toujours bornée.

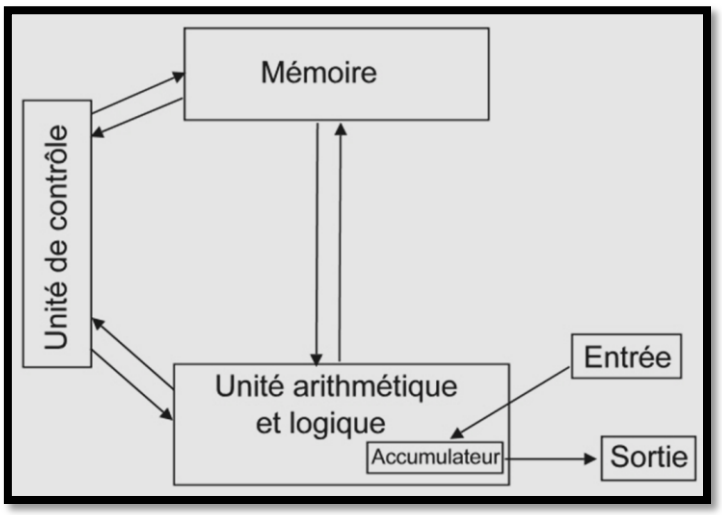


Figure 16: Machine de Von-Neumann. Source Wikipedia traduction de l'anglais.

Toute les machines actuelles sont basées en tout ou partie sur cette architecture composée d'une unité arithmétique et logique (appelée unité de traitement capable d'effectuer les opérations de base), d'une unité de contrôle (chargée du séquençage des opérations), d'une mémoire (contenant le programme et les données sur lequel les opérations sont effectuées) et enfin, d'un dispositif entrée/sortie permettant de communiquer avec d'autres machines.

3.2.2. La classification de Flynn

Il s'agit de la taxonomie proposée par Michael Flynn dans les années 60 (Flynn, 1966) pour classifier les différentes architectures parallèles suivant le nombre d'instructions concurrentes et du flux de données présents dans l'architecture (Flynn et Rudd, 1996). D'après Flynn (Flynn, 1972), il existe 4 catégories:

- **SISD** (Single Instruction Single Data Stream): il s'agit de la machine de Von-Neumann où aucun parallélisme n'est mis en jeu. C'est le cas des machines uni-processeur.
- **SIMD** (Single Instruction Multiple Data Stream): C'est un ordinateur qui introduit le parallélisme au niveau de la mémoire. C'est le cas des processeurs vectoriel ou des processeurs graphiques.
- **MISD** (Multiple Instruction Single Data Stream): C'est un ordinateur qui traite une donnée par plusieurs unités de calcul. Dans la pratique, ce type d'architecture n'est pas utilisé sauf dans les systèmes de tolérance de panne comme pour les navettes spatiales.

- **MIMD** (Multiple Instruction Multiple Data Stream): C'est un ordinateur qui est composé de plusieurs processeurs qui s'exécutent simultanément sur plusieurs données. C'est le cas des architectures distribuées (comme les Symmetric MultiProcessors (SMP)) adaptées pour faire du calcul parallèle.

3.2.3. Classification par type de mémoire

On distingue deux types de machines lorsqu'on s'intéresse à la gestion de la mémoire pour les systèmes MIMD. La Figure 17 récapitule les différentes architectures:

- **Machines à mémoire partagée ou Shared Memory (SM)**: c'est une mémoire accédée simultanément par plusieurs programmes. Il s'agit d'un bloc de RAM (Random Access Memory) accessible par plusieurs CPUs (Central Processing Units). On différencie trois classes: Uniform Memory Access (UMA), Non Uniform Memory Access (NUMA) et le COMA (Cache Only Memory Access) qui permet l'accès à la mémoire à travers un cache.
- **Machine à mémoire distribuée ou Distributed Memory (DM)**: c'est une architecture qui associe à chaque processeur sa propre mémoire. Connectées par un réseau dédié, ces mémoires distribuées exécutent chacune une partie du problème. Le calcul massivement parallèle ou Massive Parallel Processing (MPP) et le cluster computing (ou Cluster Of Workstations COW) utilisent cette architecture.
- **Machine à mémoire virtuelle partagée ou Virtual Shared Memory (VSM)**: c'est une mémoire qui est physiquement distribuée mais chaque processeur peut accéder aux mémoires spécifiques des autres processeurs.

D'autres architectures telles que le superscalar (architecture facilitant le SMT (Simultaneous MultiThreading)), le supercomputer (architecture basée sur le processeur vectoriel), les processeurs multi-cœurs (Dual Core, Quad Core et Tetra Core), les multi-processeurs cellulaires (Cell Multi-processors Array) – une technologie émergente – et les multiprocesseurs existent également. Dans la suite, un intérêt particulier sera porté aux architectures SMP, cluster et Grilles de calcul pour leur utilité pour le calcul haute-performance.

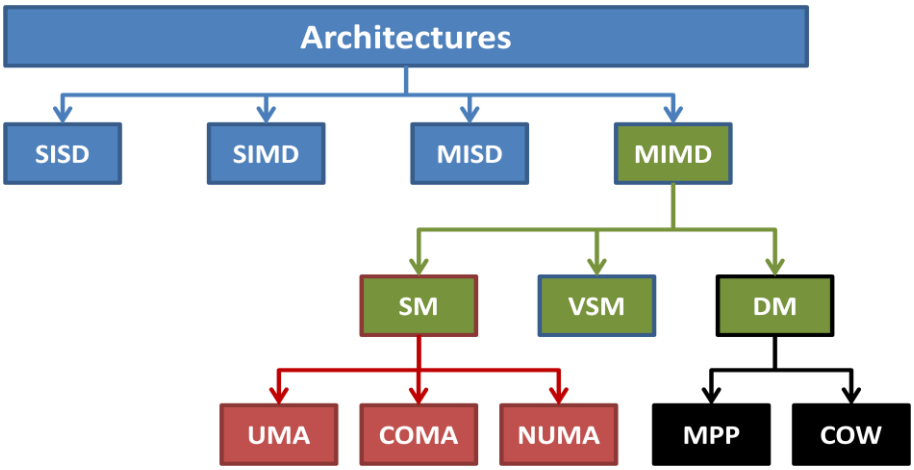


Figure 17: Classification des architectures des ordinateurs. Les acronymes sont donnés dans le texte plus haut.

3.3. Symmetric MultiProcessor

Il s'agit d'un système basé sur une architecture multiprocesseur (voir section II.3.1) qui fait appel à une mémoire partagée (SM) sur une architecture NUMA. Chaque processeur (CPU) peut exécuter une tâche indépendamment des autres processeurs de la machine SMP. Par conséquent, la performance d'un programme peut être augmentée efficacement grâce à ce système. Une telle architecture doit être supportée par le système d'exploitation installé sur un matériel approprié (Linux, Unix et Windows). L'accès se fait par un bus système commun (voir Figure 18) connecté aux entrées/sorties (I/O) du système. L'avantage de cette architecture est son extensibilité. En effet, des processeurs additionnels peuvent être rajoutés à l'architecture pour augmenter sa capacité. Il est également possible de remplacer un processeur par un autre processeur plus performant tant que le système d'exploitation le supporte. Il est important de distinguer la performance du système et la performance du programme exécuté sur ce système. Evidemment, un programme qui mettrait 20 heures à s'exécuter sur un PC monoprocesseur s'exécutera en environ 20 heures sur un SMP ayant le même type de processeur puisqu'il utilisera un seul processeur.

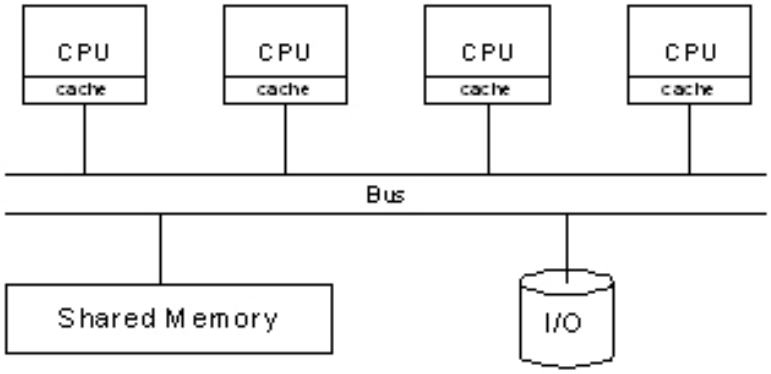


Figure 18: Architecture basique d'un SMP (Zabatta et Ying, 1998).

Le travail alors consiste à reprogrammer l'algorithme original pour qu'il s'exécute en plusieurs tâches distinctes (pouvant communiquer ou non entre elles). Le processus de transformation du code implique une reformulation, un partitionnement, un mapping et un ré-ordonnancement des données qui sont parfois complexes lorsqu'il s'agit de programmes bioinformatiques (Bokhari et Sauer, 2006). Les processeurs qui supportent le SMP sont nombreux: AMD Athlon MP, Intel Xeon, SGI/MIPS, IBM PowerPC etc.

La parallélisation des programmes informatiques peut être faite en utilisant des APIs de communication par messages (Message Passing). Elles peuvent être classifiées en deux sous-groupes d'après Hariri et Parashar, (2004):

- **L'approche de parallélisation basée sur le matériel**: cette approche vise à construire des architectures matérielles permettant au mieux de réduire la latence de communication et assurer un haut débit. Les technologies les plus connues sont NECTAR, SHRIMP ou Memory Channel.
- **L'approche de parallélisation basée sur le logiciel**: cette approche est la plus largement utilisées et consiste à incorporer des nouvelles techniques (par exemple, déploiement des techniques adaptatives, multitâches, services des middlewares…) dans des outils de communication par message existants ou à affiner les performances de certaines parties critiques de communications aux bas-niveaux (par exemple les pilotes de certains composants…) pour les réseaux rapides existants. Dans cette approche on peut distinguer trois catégories :
    - **La parallélisation standard ou « par socket »**: basée sur l'envoi direct de message entre deux processus sur la même machine ou sur deux machines différentes. C'est la technique la plus populaire utilisée sur les machines à

mémoire partagée. On peut citer MPI (Message Passing Interface) (Darling et al., 2003), OpenMP (Open Multi-Processing) (Chapman et al., 2007), PVM (Parallel Virtual Machine), UPC (Unified Parallel C) et p4 (C et Fortran).

- **Le multithreading ou le multitâches**: cette technique est capable d'ajouter une surcouche logicielle permettant de traiter le multitâche sur des APIs standards. On peut citer TPVM (Threads-oriented PVM), LPVM (Lightweight-process PVM) ou encore Pthreads (POSIX Thread).
- **Les middlewares**: ce sont des techniques consistant à modifier des APIs de communication par message pour utiliser des services particuliers de certains middlewares comme Panda et Nexus pour augmenter la portabilité du code.

Il existe également d'autres catégories axées plutôt sur une parallélisation haute performance en remplaçant les communications standards par des outils avec des communications haute performance comme ATM API, Active Message (AM), Fast Message (FM), Fast Sockets ou encore Network Characterization Service (NCS). D'autres API comme PBS (Portable Batch System) permettent de fonctionner avec une surcouche supplémentaire à l'image d'un middleware permettant la supervision des tâches parallèles indépendantes sous formes de jobs.

Bien qu'il soit possible de travailler sur des machines SMP pour diminuer les temps de calcul de certains problèmes informatiques, les systèmes d'exploitation ont leurs limites pour une seule machine lorsqu'on travaille sur des problèmes ayant besoin de temps de calcul très important. Des explications sont résumées dans les points suivants:

- http://www.freepatentsonline.com/6516442.html « *A first problem with the above-described traditional SMP system is that the serial availability of the bus limits the scalability of the SMP system. As more processors are added, eventually system performance is limited by the saturation of the shared system bus.* »
- http://www.nacad.ufrj.br/sgi/007-4639-009/sgi_html/ch01.html « *Swap space is used for temporarily saving parts of a program when there is not enough physical memory. The swap space may be on the system drive, on an optional drive, or in a file system. To avoid swapping, try not to overburden memory. Lack of adequate swap space limits the number and the size of applications that can run simultaneously on the system, and it can limit system performance.* »
- Kranz, (2008) dit: « *The next technological barrier we'll come up against is the limit of physical memory addressing for the processors. But with 64-bit architecture capable of addressing up to 16 ExaBits (10^{18} bits) of memory, this is not a barrier we'll bumping-up against anytime soon.* »

Une alternative à ces systèmes est l'architecture de cluster qui représente une collection d'ordinateurs (homogènes) inter-reliées par un réseau haut-débit fonctionnant comme un supercalculateur.

3.4. Ferme de calcul ou « Computing Cluster »

Appelée aussi « grappes de serveurs » ou encore de « multi-ordinateurs », l'architecture cluster consiste à regrouper plusieurs machines indépendantes appelées « nœuds » pour fournir une puissance de calcul plus importante que les architectures SMP et permettant de gérer la disponibilité des ressources (CPU, mémoire, disque). Les différents nœuds communiquent à travers un réseau local (Local Area Network (LAN)) haut-débit pour garantir un transfert rapide des données et partagent en général un disque de stockage commun. Il existe deux principaux types d'architecture de ferme de calcul (Trelles, 2000):

- **Les COWs**: regroupement d'ordinateurs hétérogènes connectés par un réseau haut-débit.
- **Les Beowulf**: ensemble homogène de machine Linux connecté par un réseau local (www.beowulf.org). C'est l'architecture la plus répondue grâce à son coût minime.

L'architecture est représentée par la Figure 19 qui en donne un schéma simple composé de plusieurs nœuds dont un nœud appelé « maître » (Master) et plusieurs nœuds appelés « esclaves » (Slaves).

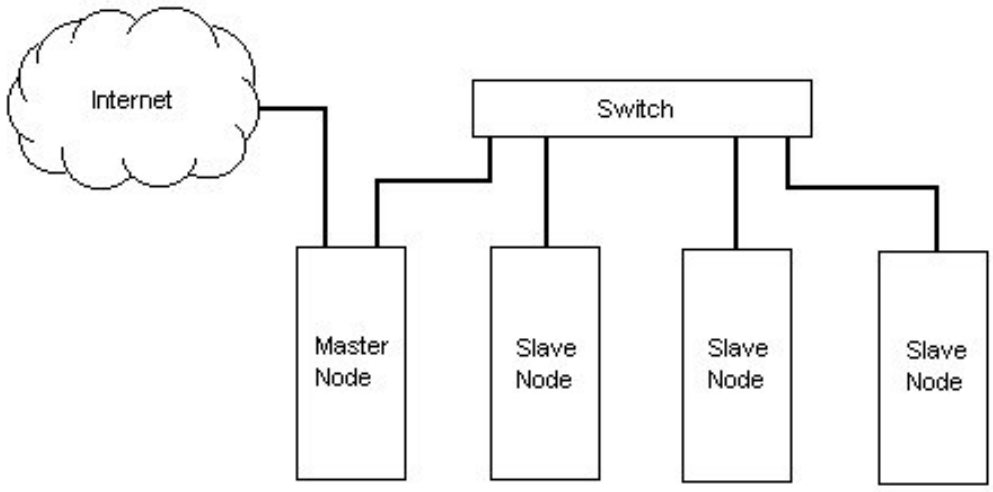


Figure 19: Schéma général d'un Cluster (Beowulf).

La quasi-totalité des clusters HPC supportent les librairies de calcul parallèle (voir section II.3.2) et offrent une extensibilité maximale. On peut citer les avantages d'adopter une architecture cluster suivants:

- ✓ Coût réduit des éléments du système comparé aux SMP.
- ✓ Puissance de calcul souvent liée à un budget matériel informatique qui est souvent plus avantageux que d'autres solutions.
- ✓ Amélioration des technologies de communication grâce à son utilisation intensive par ces architectures.
- ✓ Extensibilité qui n'est pas offerte par les architectures MPP ou SMP.
- ✓ Disponibilité qui permet à un système de continuer de fonctionner lorsqu'un nœud tombe en panne.

Dans le cadre des applications bioinformatiques nécessitant un temps de calcul important, ces architectures peuvent être très utiles (Petzold et al., 2006). Mais avec la croissance continue des données génomiques et l'explosion des quantités de séquences des banques de données, le choix est orienté vers le calcul distribué qui est la continuité logique du calcul parallèle et qui consiste à regrouper plusieurs fermes de calcul physiquement distribuées pour en tirer une puissance de calcul maximale. On parle de calcul à grande échelle. Aujourd'hui la ferme de calcul numéro 1 au top 500 (www.top500.org) des supercalculateurs mondiaux utilise un réseau avec un débit de 1 téraoctet par seconde pour l'accès au système de fichiers.

3.5. Les grilles de calcul

3.5.1. Historique

En 1969, la notion de grille de calcul a été introduite pour la première fois par Leonard Kleinrock en imaginant une infrastructure de grilles de calcul: « *Nous verrons probablement l'expansion des outils de calcul comme les réseaux électriques et les réseaux téléphoniques qui iront jusqu'à chaque maison et chaque bureau* ». Cette vision a été peu à peu implémentée durant les années suivantes par la création de centres de calcul partagés par plusieurs utilisateurs et intégrant plusieurs ressources informatiques et grâce au développement des systèmes d'exploitation multi-utilisateurs, des réseaux (terminaux distants), des architectures parallèles, des standards informatiques et enfin d'Internet (Laszewski, 2005). Au début des années 90, le terme grilles est devenu de plus en plus populaire après l'apparition des méta-ordinateurs (introduit par Larry Smarr en 1987 et désignant « *un ensemble d'ordinateurs regroupés à l'aide d'une technologie à la pointe et vus par l'utilisateur comme une seule machine de calcul* »).

En 1999, le terme « grille » est pour la première fois formalisé et défini dans (Foster et Kesselman, 2004) par: « *A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities* ». Une série de workshop et de forum dédiés aux rencontres de la communauté scientifique internationale autour de cette technologie émergente était le point de départ de la révolution informatique du 21ème siècle avec un catalyseur important, Internet. Au début des années 2000, le défi était de proposer une infrastructure qui offre des services pour l'accès aux ressources de grille en proposant différents protocoles de communications, des interfaces d'application et des kits de développement. La grille était – et est encore aujourd'hui – confrontée aux problématiques posées par la variété des technologies et services qu'englobe une telle architecture générale (voir Figure 20). La gestion d'une telle complexité se fait par le développement de logiciels de grille qui représente un kit de librairie de communication, de gestion de ressources, d'informations et d'authentification appelé aussi « middleware » tel que GT (Globus Toolkit) de la Globus Alliance (Foster, 2006).

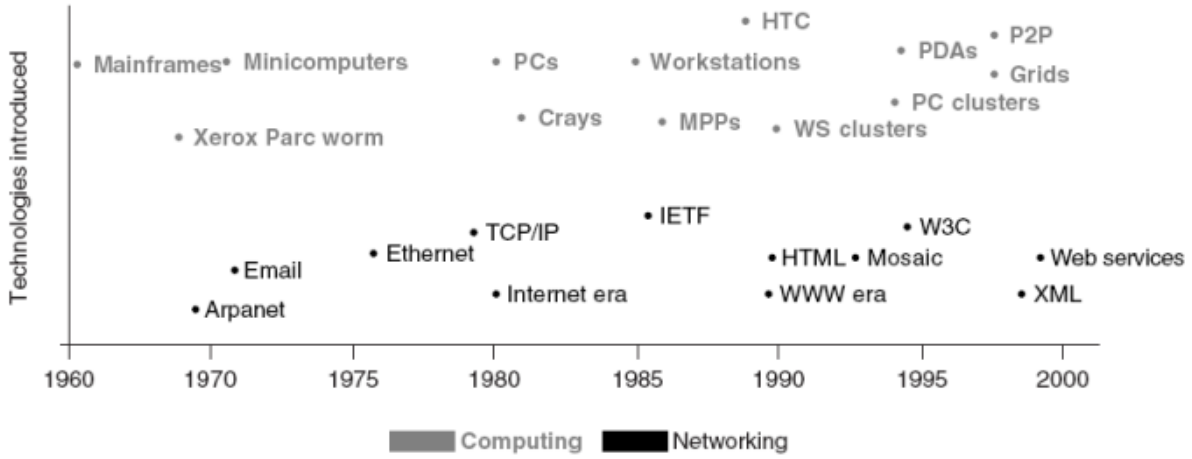


Figure 20: Evolution des technologies de l'information (Chetty et Buyya, 2002).

Les grilles représentent un ensemble de machines (multiprocesseurs) ou clusters localisés dans différents domaines administratifs distribués à travers plusieurs départements et entreprises ou distribués sur internet (Gentzsch, 2002).

L'intérêt porté à cette nouvelle technologie a amené les grandes institutions à développer leurs propres projets autour de la grille: IPG (information Power Grid) de la NASA, the Alliance Grid (Ferguson et Towns, 2001), le programme ASCI (Accelerated Strategic Computing Initiative) de la gestion des ressources informatiques sur différents sites abritant des armes nucléaires, NPACI Grid (National Partnership for Advanced Computational Infrastructure) (npacigrid.npaci.edu), DOE Science Grid (www.doesciencegrid.org), EU DataGrid (eu-datagrid.web.cern.ch/eu-

datagrid) et maintenant EGEE (Enabling Grid for E-sciencE) et TeraGrid (www.teragrid.org). On trouve un large panel des différents projets sur le site internet (www.enterthegrid.com) initié par EnterTheGrid et Primeur Magazine. Les grilles de calcul émergent alors comme une architecture de calcul haute-performance à grande échelle et d'une importance majeure et joue un rôle primordial dans tous les domaines scientifiques (chimie, biologie, physique, médecine, génie civil, etc.) par le développement de nouvelles applications innovantes et le partage de ressources à travers le monde (Foster et al., 2001).

3.5.2. Topologie d'une grille de calcul

Avant tout, une grille de calcul est une architecture permettant à plusieurs communautés de partager des ressources informatiques. L'idée centrale introduite par la grille est la notion d'organisation virtuelle (VO Virtual Organization) qui signifie un ensemble de personnes, d'institutions ou d'organismes qui ont un but commun dans leur utilisation de la grille. On peut citer la plus connue des organisations virtuelles: LHC (Large Hadron Collider) initiée par le CERN (Organisation Européenne pour la Recherche Nucléaire) qui permet d'analyser des quantités extraordinaires de données produites par le LHC. L'intérêt d'une VO est de proposer un accès simplifié aux données partagées par l'organisation ce qui évite aux utilisateurs de rapatrier des quantités de données croissantes. L'OGSA (Open Grid Services Architecture) est une architecture issue des Web Services et fournit un Framework pour la création et la gestion des services de grille et l'implémentation de la virtualisation et la gestion de l'identité. Cette architecture est composée de 3 éléments: OGSI (Open Grid Service Infrastructure), l'OGSA Services et l'OGSA Platform Models (Demchenko, 2004).

La mise en place d'une grille de calcul nécessite l'installation d'un middleware permettant de faire collaborer tous les éléments composant la grille (les personnes et les ressources). Le middleware utilisé par un grand nombre de projets de grille est GT (Globus Toolkit). La grille scientifique américaine OGS (Open Grid Science) et la grille européenne EGEE l'utilisent depuis leur création même si EGEE utilise maintenant une évolution du middleware appelé gLite.

3.5.2.1. Globus Toolkit 4

C'est le middleware de grille le plus populaire. Il a été développé en 1990 par Ian Foster et ses collaborateurs (devenue à partir de 1996 la Globus Alliance). C'est un logiciel open source établie en 1996. Il implémente les standards WSRF (Web Services Resource Framework), JSDL (Job Submission Description Language) et SOAP (Simple Object Access Protocol). On peut distinguer 5 composants élémentaires de l'architecture de GT (voir Figure 21):

- **Interface d'exécution** qui implémente différentes librairies de programmation (Python, C, et JAVA).
- **Interface de gestion d'exécution** basée sur GRAM (Grid Ressource Allocation & Management) qui assure la supervision, la gestion, l'allocation de ressources, l'ordonnancement et la coordination des programmes exécutés sur la grille et appelés Jobs.
- **Interface de sécurité** qui implémente les protocoles d'authentification et d'autorisation pour garantir une communication sécurisée avec la grille et implémente GSI (Grid Security Infrastructure).
- **Interface de Gestion des données** qui implémente GridFTP, OGSA-DAI (Data Access & Integration) et RFT (Reliable File Transfer).
- **Service d'information** constitué de composants de supervisions des ressources d'une VO basées sur MDS (Monitoring Discovery Services).

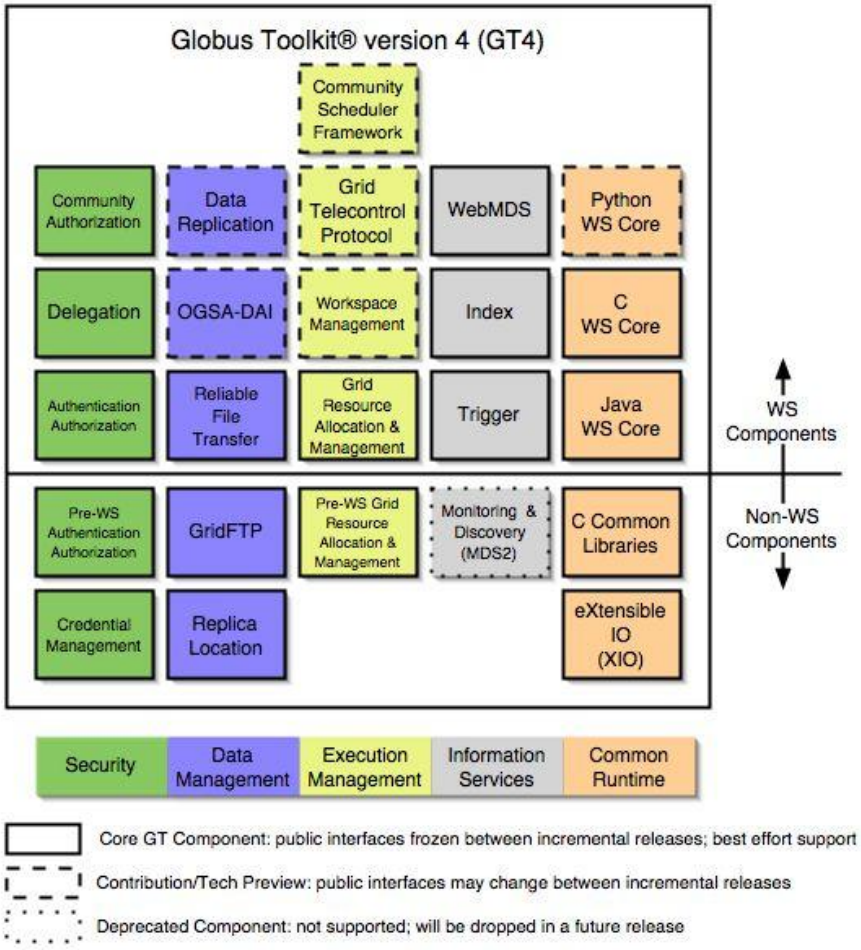


Figure 21: Architecture de GT4.

GT4 offre des services de grille bas-niveau et n'est pas capable de gérer des services plus complexes que la soumission d'un job ou la récupération d'un fichier (Foster, 2006). GT4 doit

être utilisé avec d'autres logiciels ou outils pour garantir une offre de services grilles de haut-niveau. Le problème essentiel de la grille reste la gestion de la sécurité (Hunter, 2003 ; Cody et al., 2008 ; Demchenko et al., 2008), la politique d'ordonnancement (Afzal et al., 2007 ; Caron et al., 2008), la tolérance aux fautes (Luckow et Schnor, 2008) et l'allocation de ressources (Caramia et Giordani, 2008). Un travail considérable est fait pour la standardisation des logiciels et APIs utilisés pour la gestion des grilles et la simplification de son utilisation. Wolfgang Gentzsch va jusqu'à dire dans (Gentzsch, 2008) que la grille de calcul pourrait disparaitre et laisser sa place au « *Cloud Computing* » si les grilles ne parviennent pas à être plus simple d'utilisation : *« The good news is that clouds will help grids to survive. They teach grids that in order to be widely accepted and thus sustainable, they have to be simple, user-friendly, service-oriented, scalable, on-demand, SLA-driven, with simple APIs, and so on -- just like clouds.* ».

Le « *Cloud Computing* » ou « *Internet Computing* » ou encore « *Global Computing* » est un paradigme apparu au début des années 2000 pour désigner le calcul via internet. Le projet far de ce type de calcul est SETI@HOME qui représente des millions d'ordinateurs personnels à travers la planète qui calculent ensemble des parties des données générées pour la recherche d'une intelligence extra-terrestre. Ces projets se déclinent maintenant avec BOINC[1], les APIs de Amazon Elastic Compute Cloud (Amazon EC2)[2] et Google App Engine[3]. Malgré cela, Les grilles de calcul continuent à progresser aujourd'hui dans le domaine public et privé et développent des technologies innovantes pour la gestion du calcul haute-performance à grande échelle. La grille européenne EGEE sur laquelle a été effectué le travail durant cette thèse utilise gLite, la dernière génération de middleware créé par la collaboration de 80 membres de différentes institutions et vise à fournir un Framework pour la création d'applications grille robustes et sécurisées.

3.5.2.2. gLite: Lightweight Middleware for Grid Computing

C'est un middleware développé par le projet EGEE qui reprend les standards développés par l'OGF. Il regroupe les contributions développées par d'autres projets comme LCG (LHC Computing Grid) et VDT (Virtual Data Toolkit) et ajoute des couches logicielles pour garantir des services fournis p[1]ar la grille EGEE (Laure et al., 2006):

- **Le service de gestion de sécurité** qui gère l'accès aux ressources.
- **Le service d'information et de supervision** qui permet la publication et la consultation d'informations et la supervision des jobs.

[1] http://boinc.berkeley.edu
[2] http://aws.amazon.com/ec2/
[3] http://code.google.com/intl/fr/appengine/

- **Le service de gestion des jobs** qui comprend le service d'ordonnancement, d'exécution, de traçabilité et l'installation d'applications.
- **Le service de gestion de données** qui gère les éléments d'accès et de stockage des données.
- **Le service d'accès à la grille** qui comprend les APIs nécessaires à l'utilisation de la grille.

GLite décrit et gère l'ensemble des composants de la grille EGEE. Il fournit tous les éléments nécessaires au développement d'applications grille. En effet, d'après la documentation officielle de gLite, on distingue 7 composants de l'architecture utilisée (Figure 22) en collaboration avec le WLCG (Worldwide LHC Computing Grid Project).

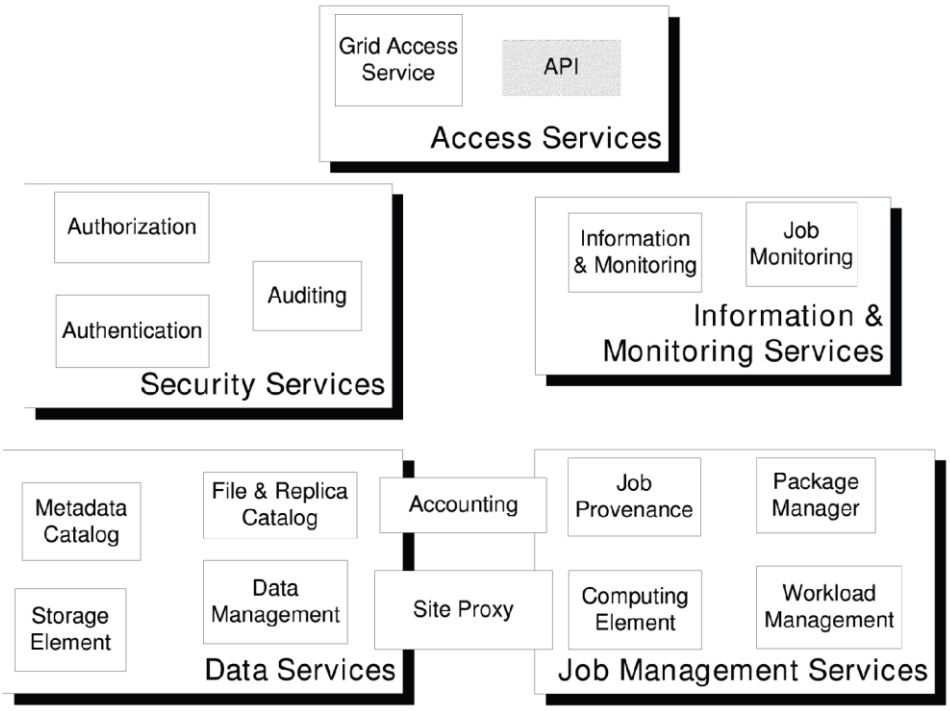


Figure 22: L'architecture de gLite.

Les composants de la grille WLCG/EGEE suivant gLite peuvent être résumés par (voir Figure 23):

- **La sécurité** gérée par le CA (Certification Authority) qui donne accès aux services de la grille à un utilisateur après avoir été enregistré dans une VO. Le GSI (Grid Security Infrastructure) fournit l'authentification et la communication par une clé publique cryptée qui sera associée à une clé privée fournit à l'utilisateur, appelée certificat proxy qui permet l'identification de l'utilisateur ainsi que ses jobs lors de leur exécution.

- **L'interface utilisateur (UI)**: C'est une machine sur laquelle l'utilisateur de la grille doit avoir un compte et installer leur certificat. L'utilisateur ne peut être identifié et autorisé à utiliser les ressources que par cette machine. Une UI permet grâce à une CLI (Command Line Interface) fournie par gLite d'effectuer les opérations élémentaires sur grille (soumission de jobs, copie de données, récupération des statuts….etc.). la grille permet le développement des applications destinées à fonctionner sur la grille.
- **L'élément de calcul (CE)**: le Computing Element représente l'ensemble des ressources présentes sur un site – EGEE regroupe plusieurs sites répartis sur 50 pays – et fonctionne comme un cluster avec une interface générique appelée GG (Grid Gate). Le cluster est composé de plusieurs WNs (Worker Node) qui représentent les unités élémentaires de calcul. Un WN contient généralement les mêmes APIs que celles installées sur une UI.
- **L'élément de stockage (SE)**: le Storage Element représente un ensemble d'API permettant de contrôler des serveurs de données, des disques de stockages massifs. Il supporte plusieurs protocoles d'accès aux données (GSIFTP, RFIO…). La plupart des sites fournissent un SE.
- **Le service d'information (SI)** : fournit des informations sur les ressources de la grille et leur statut. Les informations publiées sur la grille par le SI servent à la supervision et l'étude des performances des ressources et les informations sur les jobs sont stockées dans l'ISM (Information SuperMarket). Deux SI sont utilisés par gLite: (1) MDS (Monitoring and Discovery Service) qui implémente le schéma GLUE (Grid Laboratory Uniform Environment) du protocole LDAP (Lightweight Directory Access Protocol) sur un serveur appelé GRIS (Grid Ressource Information Server) et utilise l'index BDII (Berkeley Database Information Index) pour enregistrer et publier les données à partir du GRIS. (2) R-GMA (Relational Grid Monitoring Architecture) qui est proposé par GGF et basé sur une base de donnée relationnelle gérée par trois composants [Producer, Consumer et Registry] et utilise une partie du langage SQL (Structered query Language)
- **La gestion des données** se fait par des alias sous forme de références aux fichiers stockés sur la grille et à leurs réplicas. Les fichiers sont référencés par un GUI (Grid Unique Identifier), un LFN (Logical File Name), une SURL (Storage URL) ou un TURL (Transport URL). Les GUI et LFN identifient un fichier indépendamment de son emplacement et les SURL et TURL donnent la localisation physique du réplica d'un fichier et la façon dont on peut y accéder. Le mapping entre les GUI et les SURL est fait grâce au LFC (Logical File Catalogue) fournit par LCG.

- **Le WMS (Workload Management System)**: c'est le chef d'orchestre de la grille et est installé sur une machine appelée RB (Ressource Broker) et permet d'accepter le job soumis par l'utilisateur, l'assigne au CE approprié, enregistre sont statut et récupère sa sortie. Les jobs soumis sont décrits à l'aide du JDL (Job description Language) sous forme de fichier qui spécifie quel exécutable et quels paramètres doivent être lancés sur la grille. A l'aide d'un processus appelé « match-maker » le job est soumis au meilleur CE en fonction des informations fournies dans le fichier JDL.

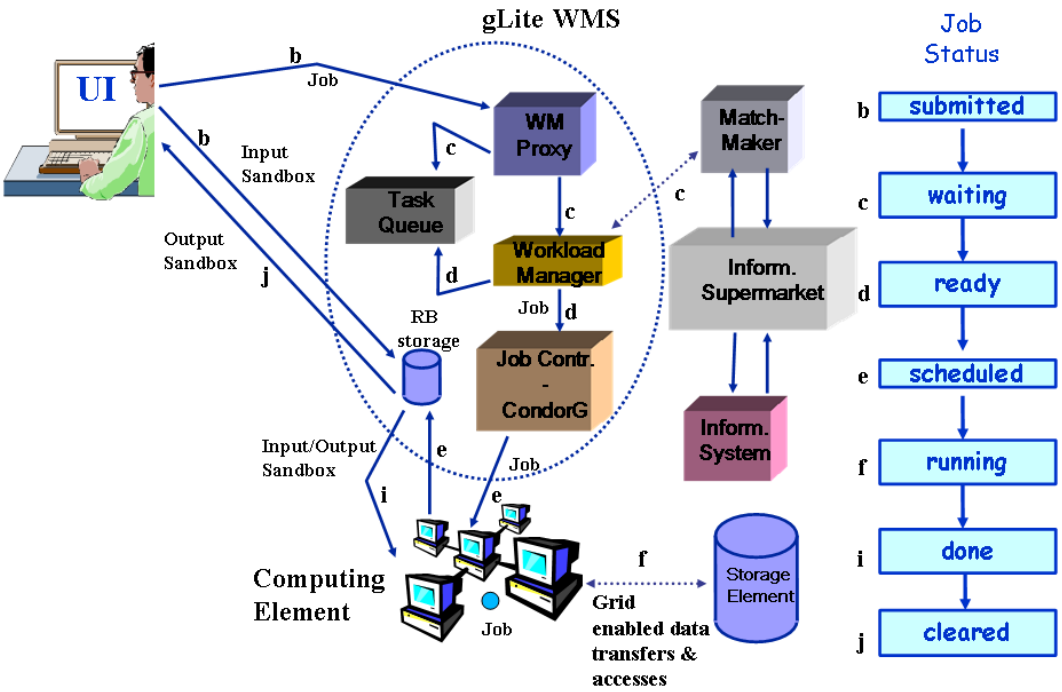


Figure 23: Les étapes d'un job avec gLite.

(Source: https://edms.cern.ch/file/722398/1.2/gLite-3-UserGuide.pdf)

GLite utilise Scientific Linux 4, une version Red Hat du système d'exploitation Linux dédiée à la grille EGEE pour tous ses composants. Aujourd'hui, un nouveau système d'exploitation basé sur Linux (XtreemOS) (www.xtreemos.org) « *a pour objectif d'enquêter et de proposer des nouveaux services qui doivent être ajoutés aux systèmes d'exploitation existants pour construire une infrastructure de grille de calcul d'une manière simple…il fournit à une grille de calcul ce que fournit un système d'exploitation classique pour une simple machine* ».

3.5.3. Les logiciels de supervisions et d'exécution de jobs

La complexité de l'utilisation de la grille due à la nécessité de connaître le middleware a poussé un grand nombre de scientifiques et professionnels à développer des outils d'exécution et

de supervision de jobs sur grille de calcul pour en faciliter la prise en main. Quelques outils émergeants sont présentés dans la suite.

3.5.3.1. GEL (Grid Execution language)

GEL (Lian et al., 2005) est un langage de description et d'exécution de jobs sur plateformes distribuées de type SMP, Cluster ou grille de type Globus en utilisant PBS, SGE (Sun Grid Engine) ou LSF (Load Sharing Facility). Il a été testé et validé pour trois applications différentes.

3.5.3.2. CoG (Commodity Grid kit)

CoG (Lazewski et al., 2000) est un outil de développement de portails destinés aux grilles de calcul qui supporte Java et Perl. Il permet une gestion haut-niveau des tâches et des données de sorties ainsi qu'une abstraction du middleware de la grille et des protocoles de transfert des fichiers.

3.5.3.3. Migrating Desktop

Les applications à la demande (Kupczyk et al., 2005) est une nouvelle approche de développement d'applications sur grille de calcul qui consiste à allouer dynamiquement les ressources systèmes pour l'intégration de systèmes HPC. Migrating Desktop est un Framework visant à mettre en pratique cette approche pour faciliter l'utilisation des applications de grille.

3.5.3.4. g-Eclipse

g-Eclipse (Wolniewicz et al., 2007) est un IDE (Integrated Developement Environment) conçu pour l'accès à la puissance des calculs des grilles à partir d'un environnement de développement JAVA. g-Eclipse (www.geclipse.org) fournit un modèle de grille de haut niveau pour manipuler les objets de grille (jobs, fichiers…) indépendamment du middleware. Des instances de ce modèle sont implémentées et fonctionnelles pour : gLite, GRIA (www.gria.org), une grille pour l'industrie orientée service, et les services de stockage S3 (Simple Storage Service) et de « cloud computing » EC2 (Elastic Compute Cloud) de Amazon. Une instance du model de grille de g-Eclipse pour la soumission de jobs de calcul sur fermes de calcul supportant PBS (Portable Batch System) est en cours d'implémentation.

3.5.3.5. Ganga

Ganga (Egede et al., 2005) est une application implémentée en Python qui permet la définition, la génération et la gestion de jobs pour des environnements distribués. Ganga fait abstraction de la plateforme d'exécution cible (processeurs de calcul locaux, Cluster ou grille de calcul). Ganga oblige chaque job à spécifier l'application à lancer, l'environnement cible, et les données d'entrée et de sortie. Optionnellement, il est possible de définir des fonctions de découpage des fichiers d'entrée et de regroupement des fichiers de sortie

3.5.3.6. Proactive

Proactive (Baduel et al., 2006) permet de faciliter l'exécution d'applications distribuées et notamment sur grille de calcul. Cependant, il peut être considéré comme un middleware écrit en Java, qui prend en charge les phases de développement, de composition et de déploiement d'applications distribuées. Proactive permet le développement d'applications distribuées en java. L'utilisation de Java permet de masquer l'hétérogénéité des environnements d'exécution ainsi que l'utilisation de RMI (Remote Method Invocation) et de l'API (Application Programming Interface) introspection java. Une application distribuée Proactive est composée d'objets actifs. Les communications entre les objets actifs se font via des appels de méthodes distantes asynchrones en utilisant les RMI et le mécanisme de communication « wait-by-necessity » Proactive permet de plus de déplacer les objets actifs entre les machines virtuelles java participant au calcul distribué et des communications de groupes vers des objets actifs de même type.

3.5.3.7. JPPF (Java Parallel Processing Framework)

JPPF (Launey et Pazat, 1997) est une plate-forme pour calcul parallèle entièrement écrite en Java qui permet l'exécution d'applications parallèles sur Cluster ou sur grille de calcul en utilisant un module d'accélération d'exécution par ordre de magnitude.

En plus des outils décrits précédemment, il existe plusieurs autres outils permettant l'abstraction des APIs de la grille en proposant – comme pour chacun des logiciels présentés ici – des surcouches logicielles permettant à l'utilisateur de se passer de l'écriture du code dans un langage donné comme par exemple GRIDRPC (Seymour et al., 2004) ou DIET (Distributed Interactive Engineering Toolbox) (Amar et al., 2006).

Enfin, la grille de calcul est aujourd'hui un outil de calcul distribué (parallèle) à part entière permettant le développement d'application gourmande en temps de calcul. L'intérêt que porte le monde pour cette architecture démontre son utilité incontestable dans la résolution de problèmes

de plus en plus complexes qu'on ne pouvait pas imaginer résoudre auparavant. L'un des domaines qui profite du développement à grande échelle des grilles de calcul est la bioinformatique (Deprez et Vernois, 2005 ; Caron et al., 2008 ; Bertis et al., 2008). La partie suivante propose un aperçu des avancées réalisées dans ce domaine grâce aux grilles de calcul.

3.6. Les grilles de calcul pour les puces à ADN

Certaines applications bioinformatiques sont gourmandes en termes de temps de calcul à cause de la complexité de leurs algorithmes traitant une quantité importante de données. Les grilles de calcul sont une solution à ces questions dans la mesure où elles permettent d'accélérer le calcul et de partager les données sur des supports de stockages suffisamment importants (Jacq et al., 2004). Les grilles de calcul ont muri considérablement et apportent désormais des solutions concrètes aux problématiques réelles haut-débits des Sciences de la Vie (Konagaya, 2006). En effet, une puissance de calcul considérable est nécessaire par exemple pour la simulation des structures secondaires des molécules, le criblage (docking) moléculaire et l'interaction spatio-temporelle des molécules. Les applications de puces à ADN rentrent dans le cadre de ces applications complexes ayant besoin d'une puissance de calcul et d'un espace disque très importants (Maglogiannis et al., 2007). L'orientation vers des web services basés sur les grilles de calcul simplifie l'accès et le partage des données de biopuces et ouvre des perspectives intéressantes pour les biologistes qui souhaitent mutualiser leurs expériences (Beltrame et al., 2007). De nombreuses applications pour le traitement des données de biopuces ont été développées depuis l'existence des grilles. Quelques outils sont présentés dans les paragraphes suivants. Ces applications concernent uniquement la partie en aval du processus de conception de biopuces : le traitement d'images.

3.6.1. Exemples d'applications pour biopuces sur grilles de calcul

Dans la littérature, il n'existe pas beaucoup d'applications implémentées sur les grilles de calcul et qui sont dédiées à la résolution de problématiques liées aux puces à ADN. Néanmoins, on peut citer les quelques architectures développées pour le traitement des images de biopuces.

3.6.1.1. GEMMA (Grid environment for microarray management and analysis)

GEMMA (Beltrame et al., 2007) est un portail web qui « …*is planned to provide shared, standardized and reliable tools for managing and analyzing biological data related to bone marrow stem cell cultures, in order to maximize the results of distributed experiments…A set of modular and independent applications may be published on the portal, and either single algorithms or a combination of them might be invoked by the user, through a workflow strategy. Services may be implemented within an existing Grid computing*

infrastructure to solve problems concerning both large datasets storage (data intensive problem) and large computational times (computing intensive problem) »

3.6.1.2. Grid Portal for microarrays

La solution développée par (Porro et al., 2007) est un portail de gestion, et d'analyse des puces à ADN : « *A state-of-art Grid portal has been implemented in order to hide the complexity of framework from end users and to make them able to easily access available services and data. The functional architecture of the portal is described. As a first test of the system performances, a gene expression analysis has been performed on a dataset of Affymetrix GeneChip® Rat Expression Array RAE230A, from the ArrayExpress database. The sequence of analysis includes three steps: (i) group opening and image set uploading, (ii) normalization, and (iii) model based gene expression (based on PM/MM difference model)* ».

3.6.1.3. XPS (eXpression Profiling System)

L'application présenté dans (Stratowa, 2003) est un système de profiling de l'expression de gènes d'expériences de puces à ADN: « …*a functional prototype system, called XPS - eXpression Profiling System, which can be considered to be an alternative to the BioConductor project. The current implementation handles efficient storage of Affymetrix GeneChip schemes, gene annotation and data, and the pre-processing, normalization and filtering of GeneChip data.* ».

3.6.2. Le problème de conception de sondes pour biopuces à ADN sur grille de calcul

Comme nous l'avons vu précédemment, la conception de sondes est une tâche difficile et couteuse en terme de temps de calcul et d'espace de stockage et de mémoire et cela est dû, à la fois, à la taille des données manipulées, et, à la complexité des algorithmes employés (Oroguchi et al., 2005). En ajoutant la complexité de l'utilisation des technologies de grilles de calcul, le problème devient plus complexe. Des approches de simplification par décomposition du problème sont nécessaires pour l'aborder. Le programme bioinformatique le plus consommateur de ressources dans une conception d'oligonucléotides est BLAST. Il est également largement utilisé par la communauté scientifique pour la recherche de similarité de séquences biologiques. D'autres programmes qui interviennent dans la conception comme RepeatMasker (complexité des séquences), Mfold (structures secondaires des séquences), Clustalw (alignement multiple de séquences) ou autres nécessitent également d'importantes ressources. Plusieurs travaux de distribution de BLAST sur grille de calcul ont été réalisés ces dernières années.

3.6.2.1. BLAST sur grille de calcul

La parallélisation du BLAST sur plates-formes parallèles types SMP et Cluster a été largement étudié (Wu, 2006). La nature extrêmement parallèle du BLAST facilite sa distribution sur plates-formes hétérogènes de type grille de calcul (Afgan et al., 2006). La parallélisation peut être réalisée en fragmentant le fichier « query » contenant les séquences à comparer ou en fragmentant la base de données de comparaison. Dans le premier cas, les jobs créés sont totalement indépendants. Une étape de réassemblage des résultats des différents fragments est alors nécessaire à la fin du calcul. Dans le deuxième cas, une étape de calcul de l'e-value globale est nécessaire pour obtenir le même résultat. Le Tableau 1 récapitule quelques versions de BLAST adaptées pour les grilles de calcul. Les applications peuvent être un service web, un package de scripts ou un une application client-serveur indépendante ou faisant partie d'un ensemble de plusieurs services bioinformatiques. La plupart des BLASTs font partie de pipelines complets d'applications bioinformatiques (Raih et al., 2005).

Tableau 9: Les BLAST sur grille de calcul dans la littérature.

Références	Découpage de l'entrée	Fragmentation de la base	Réplication	Gestion des erreurs	Type de l'application
Konishi et al. 2003	Oui	Oui	Non	Oui	CS
Kumar et al., 2004	N.C	Non	Oui	N.C	WS
Bayer et al., 2005	Oui	Oui	Non	Non	WP
Krishnan et al., 2005	Oui	Non	Oui	Oui	Script
Dowd et al. 2005	Oui	Non	Non	Oui	Standalone
Carvalho et al. 2005	Oui	Non	Oui	Oui	Script/UI
Trombetti et al.,2007	Oui	Non	Oui	Oui	WP
Sun et al., 2007	Oui	Non	Non	Oui	CS
Afgan, 2006	Oui	Non	Oui	Oui	N.C
Mirto et al, 2008	Oui	N.C	N.C	Oui	WS

N.C: Not Communicated -- CS: Client/Serveur -- WS: Web service -- WP: Web portal

Dans (Chen et Schmidt, 2005) une nouvelle approche basée sur l'équilibrage dynamique de charge appelée « *dynamic-scheduler* » est utilisée pour la soumission des fragments sur grille de calcul. L'intérêt de telles applications est de proposer une solution haute-performance de BLAST utilisable seule ou dans d'autres applications bioinformatiques telles que la conception de sonde ADN.

3.6.2.2. AMGA : Bases de données distribuées sur grilles de calcul

La conception d'oligonucléotides pour les biopuce à ADN est une tâche qui dépend toujours de la base de données de référence utilisée pour la recherche d'unicité des sondes potentielles. Avec l'augmentation exponentielle des séquences dans les banques de données internationales, il est devenu indispensable d'inventer un système de bases de données distribuées. AMGA (ARDA Metadata Grid Application) (Koblitz et al., 2008) est une application originale proposée par EGEE dans le cadre de son projet gLite. Elle représente un « metadata catalogue » qui fournit un accès à un metadata pour les fichiers stockés sur grille de calcul, et permet d'accéder aux bases de données relationnelles avec un système de sécurité basé sur un système de gestion des organisations virtuelles. AMGA est distribuée avec les APIs de gLite et est composée d'une partie « serveur » et d'une partie « client » (Figure 24). La partie serveur supporte différents SGBD (Système de Gestion de Bases de Données) et gère les catalogues par un système de réplications asynchrones maitre-esclave des données. La sécurité est assurée grâce à une authentification par mot de passe, un certificat grille X509 et un certificat VOMS (Virtual Organization Membership Service) anonyme.

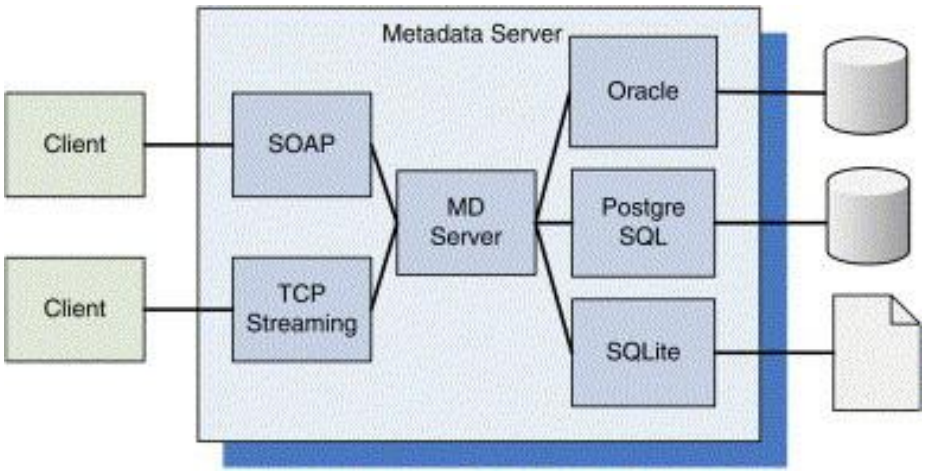


Figure 24: Les composants principaux d'AMGA.

(Source: (Santos et Koblitz, 2006))

3.7. Synthèse

Bien que les grilles de calcul soient un outil de calcul haute-performance, elles offrent aussi une alternative intéressante mais encore à perfectionner pour la gestion des données à grande échelle. A notre connaissance, aucun logiciel de conception de sonde n'a été développé sur une grille de calcul à ce jour. Seul l'algorithme de Zhu et al., 2006 pour la conception de sonde d'ARN 16S a été parallélisé sur architecture de type Cluster. Phylarray (Militon et al., 2007) profite également d'une architecture parallèle de type SMP et/ou Cluster pour la conception de sondes 16S pour biopuces phylogénétiques.

4. Conclusion

Le besoin évident de l'informatique pour manipuler les données de la biologie moderne d'une part et la recherche permanente de haute-performance due à l'explosion des quantités de données génomiques d'autre part, sont les principales causes de l'engouement sans précédent autour des technologies de l'information et de la communication. Dans le cadre de l'étude de la biodiversité microbienne par des méthodes haut-débit telles que les puces à ADN, des algorithmes complexes gourmands en temps de calcul et utilisant une quantité de données considérable sont employés pour la recherche de sondes spécifiques. L'ingénierie des modèles et les grilles de calcul en tant que technologies émergentes de l'informatique offrent des possibilités innovantes pour la résolution des problématiques de modélisation et de performance liées à ce domaine.

Aujourd'hui, le développement à grande vitesse de plates-formes distribuées de calcul haute-performance telles que les grilles de calcul a permis la résolution de plusieurs problèmes en biologie (Talbi et Zomaya, 2008). En effet, un nombre important de projets de grilles de calcul a vu le jour au cours des dernières années. La conception de sondes pour biopuces à ADN est un domaine bioinformatique où la recherche d'oligonucléotides spécifiques et sensibles est une priorité. Plusieurs algorithmes de sélection d'oligonucléotides souffrent d'une lenteur due essentiellement à la recherche de spécificités (Adebiyi, 2007). Array Designer 3.0 par exemple a préféré la performance à la spécificité en développant une nouvelle version 4.0 et perd alors en spécificité avec cette nouvelle version. OligoArray 1.0 de son coté, reste moins performant que d'autres algorithmes comme YODA (Nordberg, 2005). Certaines approches de sélection de sondes ADN sont axées sur des recherches de spécificités étendues sur des grandes bases de données ce qui augmente considérablement le temps de calcul total.

Enfin, le développement d'APIs (Application Programming Interfaces) de haut niveau (comme gLite) et de logiciels de supervisions de jobs a permis de faciliter l'utilisation de la grille de calcul qui reste malgré tout un outil nécessitant une bonne connaissance des logiciels employés.

Chapitre III : Matériels et méthodes

1. Introduction

Nous avons vu dans les chapitres précédents que la caractérisation de la biodiversité microbienne dans les environnements complexes avec les approches des biopuces à ADN représente une difficulté considérable due essentiellement à la complexité des algorithmes de sélection de sondes d'une part et à une quantité de données importante à gérer d'autre part. Il est donc important de contrôler les données disponibles dans les banques génomiques internationales qui serviront comme données de référence aux logiciels de conception d'oligonucléotides mis en œuvre. De plus, avoir des outils adaptés pour l'analyse, le traitement et la transformation de ces données devient quasiment obligatoire car le besoin de chaque expérience est différent. En effet, pour une biopuce phylogénétique par exemple, la conception se base principalement sur les séquences conservées codant pour les ARN ribosomiques 16S (procaryotes) et 18S (eucaryotes), alors que pour une biopuce fonctionnelle tout type de transcrit des communautés microbiennes étudiées est pris en compte dans la recherche des sondes. Nous présenterons dans ce chapitre un système d'information dédié aux données biologiques pour les puces à ADN.

Par ailleurs, et dans le cadre de la découverte de nouvelles espèces et la caractérisation de nouvelles voies métaboliques d'intérêt, il est également judicieux de travailler sur les séquences protéiques. Notons que certains de ces biomarqueurs peuvent également être utilisés dans le cadre d'études phylogénétiques. Une nouvelle approche de traduction inverse de séquences protéiques courtes basée sur l'Ingénierie Dirigée par les Modèles (IDM) sera présentée dans ce chapitre.

Enfin, une distribution sur grille de calcul du programme BLAST nécessaire pour la recherche d'un nombre croissant de similarité de séquences est introduite à la fin de ce chapitre.

2. Système d'information pour la conception de sondes pour biopuces à ADN

L'idée de proposer une nouvelle approche pour la conception de sondes pour différents types de biopuces à ADN est liée essentiellement à la complexité des informations relatives aux séquences biologiques que l'on retrouve dans les banques de données internationales et la quantité de données à gérer qui ne cesse d'augmenter (Figure 25). En 25 ans, le nombre de séquences présentes dans la banque de l'EMBL[1] (European Molecular Biology Laboratory) est passé de 568 à près de 100 millions de séquences. Aujourd'hui, la banque EMBL compte environ 150 millions d'entrées pour la release 97 de janvier 2009.

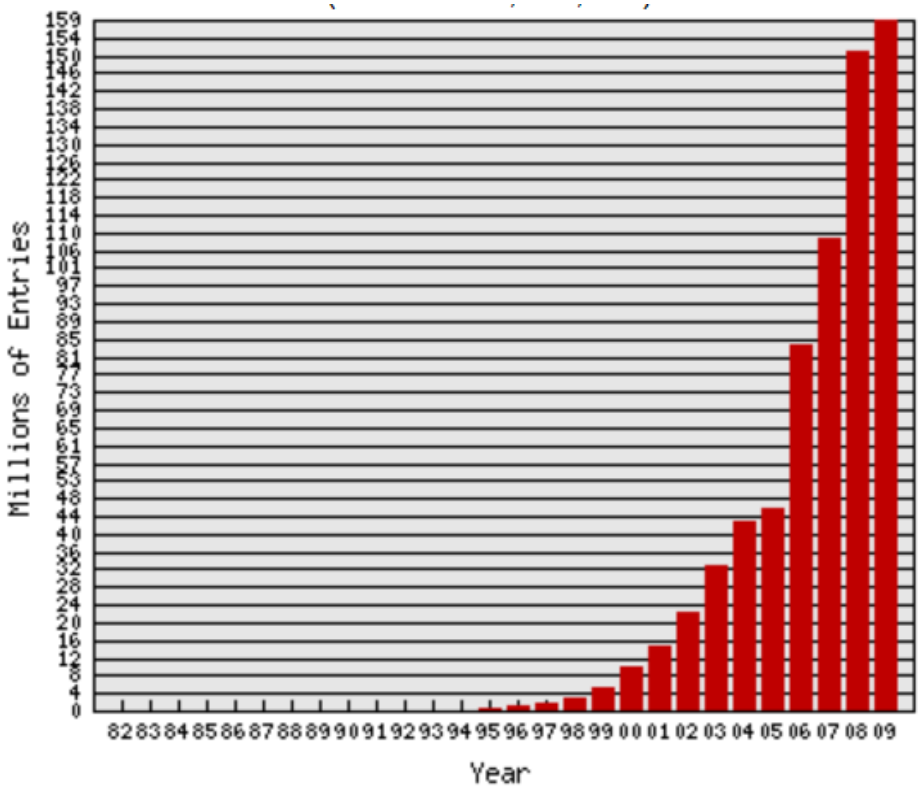


Figure 25: Nombre d'entrées dans la banque EMBL depuis 1982.

(Source: http://www.ebi.ac.uk/embl/)

Les informations sur les annotations de génomes sont parfois incomplètes ou erronées (Devos et Valencia, 2001) et demandent une vigilance de la part des biologistes et des bio-informaticiens lors de la sélection des données qui serviront pour construire les bases de données de référence contre lesquelles seront comparées les sondes oligonucléotides pour la recherche des

[1]EMBL release: ftp://ftp.ebi.ac.uk/pub/databases/embl/release

hybridations croisées potentielles. Par exemple, d'après Jones et al., (2007) le pourcentage d'erreur est à peu près de 30% pour la base GoSeqLite (http://www.geneontology.org/) due aux annotations erronnées. Les banques de séquences de 16S ne sont pas épargnées et montrent également plus de 10% de séquences chimériques sans compter les erreurs d'annotation (Ashelford, 2006).

Dans le cadre de la conception des sondes ADN, il est impératif de proposer une base de données fiable contenant des informations exactes. Plusieurs étapes complexes sont nécessaires pour construire des bases de données de qualité. Nous proposons un algorithme d'analyse et de construction de données de séquences pour la conception de sondes. Signalons que ces bases sont réutilisables pour d'autres applications biologiques (recherche de similarité, extraction de données…). Une démarche IDM pour la création d'un système d'information (base de données) est également présentée.

2.1. Algorithme d'analyse et d'extraction de données

L'algorithme proposé ici consiste à utiliser la banque de données internationale EMBL en extrayant des informations d'annotations précises des fiches EMBL des séquences biologiques. Le programme est entièrement développé en BioPerl (www.bioperl.org). Dans le cadre de l'étude de la biodiversité microbienne dans les environnements complexes, toutes les classes de données EMBL et les divisions taxonomiques qui représentent les populations microbiennes de ces environnements à savoir les champignons (FUN), les procaryotes (PRO) et les échantillons environnementaux (ENV) ont été considérées (voir section I.3.1). Le processus est composé des étapes principales suivantes.

2.1.1. Téléchargement des données

Une connexion FTP est ouverte vers le site de téléchargement des dernières mises à jour des fichiers EMBL compressés au format « .dat.gz » à l'aide du module NET::FTP de BioPerl. Ensuite, sont téléchargées toutes les fiches EMBL compressées qui correspondent aux divisions et aux classes définies au préalable. Les noms de fichiers à télécharger sont uniques et sont décrit de la manière suivante (à part les fichiers WGS de la forme « wgs_alphabet_div[_nb].dat.gz ») :

rel_class_div_nb_relnu.dat.gz

Où « class » : est l'abréviation correspondant à la classe des données ;

« div » : est l'abréviation correspondant à la division des données ;

« nb » : est le numéro du fichier de même type ;

« relnu » : est le numéro de la mise à jour (release).

2.1.2. Décompression des fichiers

Chaque fichier téléchargé est ensuite décompressé pour être traité. Il donne lieu à une fiche EMBL qui contient une ou plusieurs entrées. La taille totale des données téléchargées est de 15Go de fichiers compressés pour la version 97. Le Tableau 10 récapitule le nombre d'entrées pour chaque division taxonomique en prenant en compte toutes les classes. Après décompression, l'extension des fichiers est « **.dat** ». Ils sont identifiables par leurs caractéristiques décrits plus haut.

Tableau 10: Nombre de séquences et de nucléotides dans la base EMBL correpondants aux divisions microbiennes (version 97).

Division	Entrées	Nucléotides
ENV: Echantillons d'environnement	24 455 401	14 345 781 152
FUN: Champignons	2 816 771	4 738 183 474
PRO: Procaryotes	918 456	6 290 377 815
Total	28 190 628	25 374 342 441

2.1.3. Traitement des entrées EMBL

Cette étape consiste à extraire les informations d'annotation enregistrées dans les fiches. Une entrée correspond suivant son type à une séquence génomique, à un génome ou à une simple séquence issue d'un séquençage d'ADNc ou de produits PCR. L'algorithme s'intéresse plus particulièrement aux séquences codantes (CDS) des microorganismes appartenant aux divisions taxonomiques choisies. Pour chaque entrée EMBL, l'algorithme compte le nombre de régions codantes en repérant dans les fiches les coordonnées des CDSs s'ils existent. Dans le cas ou aucun CDS n'a été trouvé, la séquence n'est pas prise en compte. Dans le cas contraire, chaque élément « feature » correspondant à un CDS est parcouru pour en extraire les données nécessaires à l'identification de la séquence. Ces informations sont signalées par des balises « tag » décrivant la molécule en question. Les éléments correspondant à « source » sont parcourus pour en extraire le nom de l'organisme d'origine de la molécule (l'entrée) traitée et son type. Un CDSs est caractérisé par sa séquence génique, ses coordonnées sur la séquence complète de la fiche EMBL, la fonction du produit du CDS, l'identifiant protéique, et la séquence protéique correspondantes. Le CDS peut se trouver sur le brin (+) ou le brin (-) (désigné par le mot clé « complement ») d'un gène annoté. L'algorithme localise chaque CDS dans l'entrée EMBL pour en extraire les positions relatives et en cas de besoin faire une traduction de la séquence nucléique

pour obtenir la protéine correspondante. La Figure 26 représente une entrée EMBL de la division des Procaryotes WGS décrivant la séquence AAZV01000021 qui représente de l'ADN génomique et se trouve dans le fichier wgs_aazv_pro.dat.gz téléchargé précédemment. Les coordonnées du CDS indiquent les positions « start » et « end » dans la séquence nucléique signalée par « SQ » de l'entrée EMBL en cours. La fonction de la protéine est donnée par le champ « product », son identifiant est donné par le champ « db_xref » et sa séquence est donnée par le champ « translation ».

```
FT   CDS             <1..461
FT                   /codon_start=3
FT                   /transl_table=11
FT                   /locus_tag="LVAL_00026"
FT                   /product="pseudouridine synthase Rsu"
FT                   /note="COG1187 16S rRNA uridine-516 pseudouridylate
FT                   synthase and related pseudouridylate synthases"
FT                   /db_xref="GOA:A6BL18"
FT                   /db_xref="InterPro:IPR000748"
FT                   /db_xref="InterPro:IPR006145"
FT                   /db_xref="UniProtKB/TrEMBL:A6BL18"
FT                   /protein_id="EDL71127.1"
FT                   /translation="AQHRLLDPKFGKSRAYWVQVERVPDEASLQRLRDGVTIRQYRTRP
FT                   ARVRRLAIAPDLPPRHPPIRFRKNVPTCWLEMSLTEGRNRQVRRMTAAVGHPTLRLVRV
FT                   AVENLTLDALQPGQWRDLTLAEVSQLQKRCGLSAVASRSGDRNRARYN"
XX
SQ   Sequence 884 BP; 181 A; 300 C; 216 G; 184 T; 3 other;
     tcgctcagca ccgactcctc gatccgaaat ttggtaagag ccgcgcgtac tgggtacaag        60
     tcgaacgggt tcccgacgaa gccagcttac agcgcctgcg cgacggcgtc acgatccgtc       120
     agtaccgtac gcgacccgcc cgtgttcgac ggttggcgat cgcccccgac ttaccgccga       180
     gacatccccc gattcgcttc cgtaaaaacg ttccgacctg ctggctggaa atgagcctca       240
     cagaagggcg aaaccgccaa gtccgacgca tgaccgccgc tgtgggacat ccgacgttac       300
     gcttggtgcg cgtcgccgtc gaaaacctaa cgctagacgc tctgcaaccg ggacagtggc       360
     gagatctcac tctcgctgaa gtcagccaac tccaaaagcg gtgtgggttg tccgccgtcg       420
     cgtcccgttc gggcgatcga aatcgcgcac gatataatta atagctttcg atcgccgtcg       480
     ccatgttggt ttgcccccac tgtcacaccc tcaatcccga aggtcgccaa tattgctcgc       540
     aatgtggggc gtcgattccc tcacacaaac cggtttggtt cgccgccatc gtctcgacga       600
     agtttaccgc ctcgaccctt cctccctatc tcgatcccaa acaacgctat cgtttcttcg       660
     atctgggaaa actcaacgag tacggcgaca tcgaaactcg cgcctacgac ctccaacccc       720
     agctcccccc ttatttcaac tctagcgtca gttcttacag cgcggacagt cacgttccta       780
     agttggcgcg gccgtactgc gatccccgct gtcaggcgaa ttccgccttt ccgcgacttc       840
     ncgatacctg ggaangcgac nactatacag tcgttctgct cgaa                        884
//
```

Figure 26: Exemple d'entrée EMBL.

2.1.4. Extraction et contrôle des séquences

Les séquences visées par l'approche implémentée par l'algorithme sont les séquences transcrites (tous les ARN messagers). L'objectif étant d'utiliser la biopuce fonctionnelle dans une approche de type métatranscriptomique. La reconstruction des séquences se fait en ajoutant aux CDS les régions non traduites – potentiellement transcrites – appelées UTR (Untranslated Region) composées des UTR 5' et UTR 3'. La longueur des UTRs (qui correspondent aux opérons pour les procaryotes) a été fixée à « utr_length » égale à 300 bases. De plus, comme pour l'extraction du CDS, les UTRs sont extraits à partir de la séquence nucléique correspondant à l'entrée EMBL en cours de traitement sur le brin portant le CDS. La Figure 27 montre un

exemple de cas particuliers qui peuvent se présenter lors de la recherche des régions transcrites mais non traduites (le CDS2 est extrait par l'algorithme). Les séquences obtenues par concaténation des CDSs et de leurs UTRs correspondants sont ensuite contrôlées par le programme en substituant les insertions erronées d'acides nucléiques inconnues lors de la procédure d'annotation par des 'N'. Les traductions correspondantes sont également vérifiées en remplaçant les acides aminés inconnus par des 'X'.

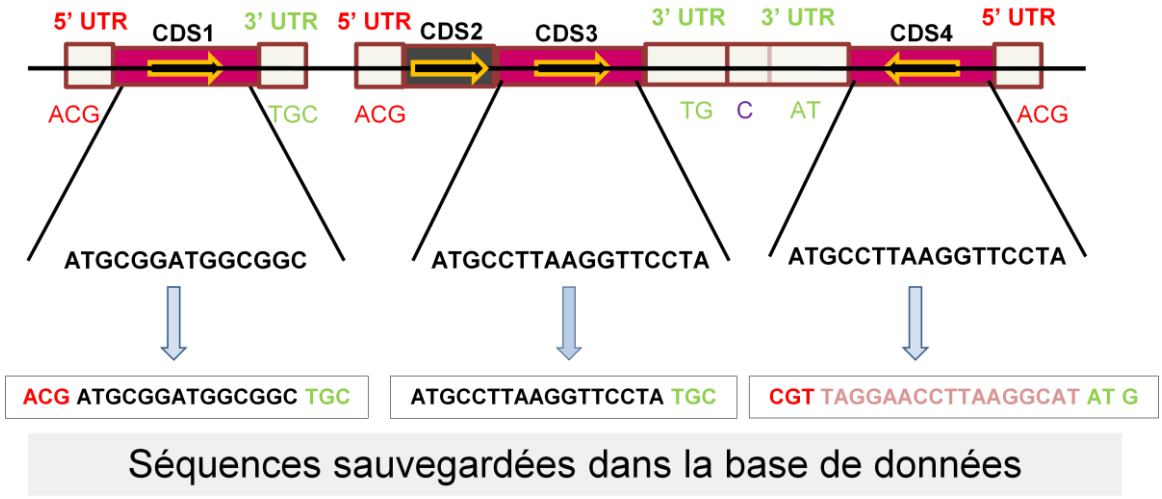


Figure 27: Extraction des séquences à partir d'une fiche EMBL.

2.1.5. Sélection et sauvegarde des séquences

La dernière étape consiste à vérifier les séquences construites dans l'étape précédente en éliminant celles qui sont courtes et qui présentent un nombre élevé d'insertion ('N' et 'X') et en tronquant celles qui présentent une terminaison 3' de mauvaise qualité c'est-à-dire admettant plusieurs 'N'. Les informations collectées par l'algorithme sont ensuite préparées pour être intégrées dans une base de données relationnelle pour une meilleure visibilité et plus de facilité d'intégration et d'utilisation. Chaque séquence nucléique est caractérisée par un identifiant, un numéro d'accession, un libellé, un CDS, une position de début, une position de fin, un type de molécule et une taxonomie. De même, une séquence protéique est caractérisée par une séquence nucléique, un identifiant, un numéro d'accession et un nom de produit. Ces informations permettront une flexibilité dans le choix des bases de données de référence pour la conception de sondes ADN spécifiques en appliquant des critères de sélection précis. La conception de sondes oligonucléotides peut être réalisée à partir des séquences nucléiques en se basant uniquement sur le CDS repéré par les positions relatives de début et de fin sur les séquences sauvegardées ou à partir des séquences protéiques en effectuant une traduction inverse pour reconstruire les séquences nucléiques potentielles correspondantes. Afin de stocker ces données, une démarche

de génie logiciel a été adoptée en utilisant SysML pour modéliser une base de données relationnelle utile pour l'intégration et la centralisation de ces données.

2.2. Conception d'une base de données relationnelle avec SysML

Depuis 1993, le journal *Nucleic Acids Research* présente chaque année dans son numéro spécial « *Database issue* » les bases de données biologiques développées par la communauté scientifique. Une collection est mise à jour tous les ans et contient toutes les bases de biologie moléculaire classées par catégorie[1]. En 2009 (Galperin et Cochrane, 2009), elle compte 1170 bases de données dont 95 nouvelles par rapport à 2008. Dans (Galperin, 2005), une liste complète des bases de données développées jusqu'en 2005 a été présentée. Depuis, tout est référencé sur le site du journal. Ces bases de données centralisent des informations utiles pour un large public de biologistes. La conception de telles bases de données nécessite souvent une phase d'ingénierie permettant de modéliser les composants logiciels du système d'information final qui sera utilisé pour exploiter les données génomiques et biologiques ciblées. Les langages les plus utilisés dans la conception de base de données est Merise (Tardieu et al., 1983) et les diagrammes Entité-Association. Le langage UML (Unified Modeling Language)[2], dans sa version 2.0 propose d'étendre le méta-modèle pour définir un vocabulaire permettant de créer de nouveaux modèles dérivées de modèles déjà existants mais qui ont des propriétés spécifiques adaptées pour des problèmes particuliers ou une utilisation plus spécialisées.

En mars 2003, l'OMG et l'INCOSE soumettent pour la première fois une requête pour proposition (RFP) concernant des spécifications « UML pour l'ingénierie des systèmes » qui est devenue un peu plus tard en mai 2006 ce qui est appelé aujourd'hui SysML (System Modeling Language). SysML est un langage de modélisation graphique généraliste pour spécifier, analyser, concevoir et vérifier des systèmes complexes qui peuvent inclure du matériel, du logiciel, des informations, des personnes, des procédures (www.sysml.org)... En particulier, le langage fournit des représentations graphiques avec une fondation sémantique pour modéliser des exigences système, des comportements, des structures, et l'intégration avec un large éventail d'outils d'analyse et d'ingénierie.

La première étape dans une démarche de modélisation à l'aide de SysML consiste à établir le diagramme de cas d'utilisation relatif au fonctionnement du système d'information. Il s'agit précisément de la base de données présentée ici et appelée ProtBase, qui a été développée pour gérer efficacement les données collectées par l'algorithme décrit dans la partie III.2.1. Elle ajoute

[1]NAR Database Collection: http://www.oxfordjournals.org/nar/database/c/
[2]UML 2.0: http://www.uml.org/

une partie concernant la gestion des oligonucléotides dans le cadre de la conception de sondes ADN. Le diagramme de cas d'utilisation fait partie des diagrammes que partage SysML avec le langage UML (voir Figure 28). Il permet de décrire les fonctionnalités du système.

Figure 28: Les diagrammes du langage SysML.

(Source: OMG SysML tutorial - 11 juillet 2006 www.omgsysml.org)

En utilisant « *Oracle JDeveloper 10 Diagrams* », un diagramme de cas d'utilisation a été réalisé. Trois types d'acteurs ont été identifiés : l'utilisateur de la base de données, l'administrateur, et les programmes de mise à jour incluant l'algorithme d'extraction et d'enregistrement des données (Figure 29). L'utilisateur peut visualiser et sauvegarder les données. L'administrateur peut modifier les données et s'occupe de la maintenance logicielle. Les programmes externes s'occupent de la mise à jour des versions et permettent d'alimenter la base de données.

Les fonctionnalités de la base de données étant établies, la deuxième étape d'une modélisation à l'aide de SysML consiste à définir les composants et les exigences du système en utilisant respectivement le diagramme de définition de blocs (Block Definition Diagram « bdd ») et le diagramme des exigences (Requirement Diagram « rd »), nouveautés proposée par SysML par rapport à UML 2. Il s'agit donc de définir dans un premier temps les entités qui composeront la base de données. L'information centrale concerne principalement les séquences correspondant aux régions transcrites disponibles dans les banques génomiques. Chaque séquence contient éventuellement des régions UTR 5' et 3' si elles existent.

Figure 29: Diagramme de cas d'utilisation de la base de données.

La séquence est composée d'un CDS et des régions UTR associées. Une séquence nucléique, provenant d'un CDS et présente dans la banque de données, donne lieu à une séquence protéique. La Figure 30 montre un diagramme de définition de blocs basé sur les spécifications précédentes. Un « bdd » permet de définir les composantes du système et leurs interactions. La modélisation a été effectuée avec Topcased [1], un outil open source de modélisation de systèmes et de logiciels.

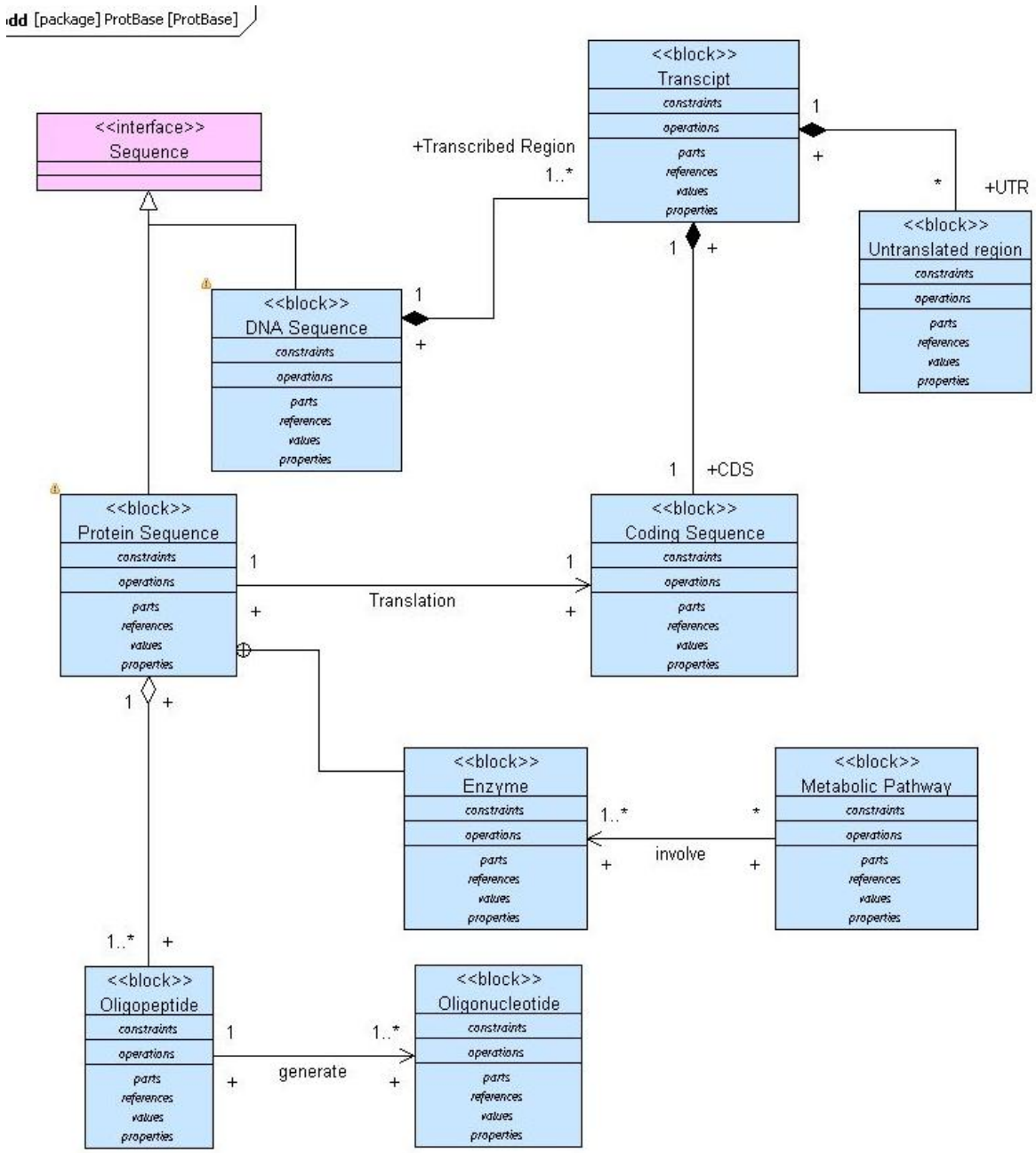


Figure 30: Diagramme de définition de bloc de la base de données.

Pour modéliser les besoins (les exigences du système), SysML offre un nouveau diagramme appelé diagramme des exigences également implémenté dans la version 2 de Topcased[1] et permet de définir des conditions sur les blocs ainsi que des relations entre ces conditions. Contrairement aux autres diagrammes de SysML qui diffèrent de UML 2.0 et qui implémentent des profils déjà définis par le langage, le diagramme des exigences est définit au niveau du méta-modèle[1]. La Figure 31 représente un diagramme des exigences pour la base de données.

[1]Topcased: http://topcased-mm.gforge.enseeiht.fr/

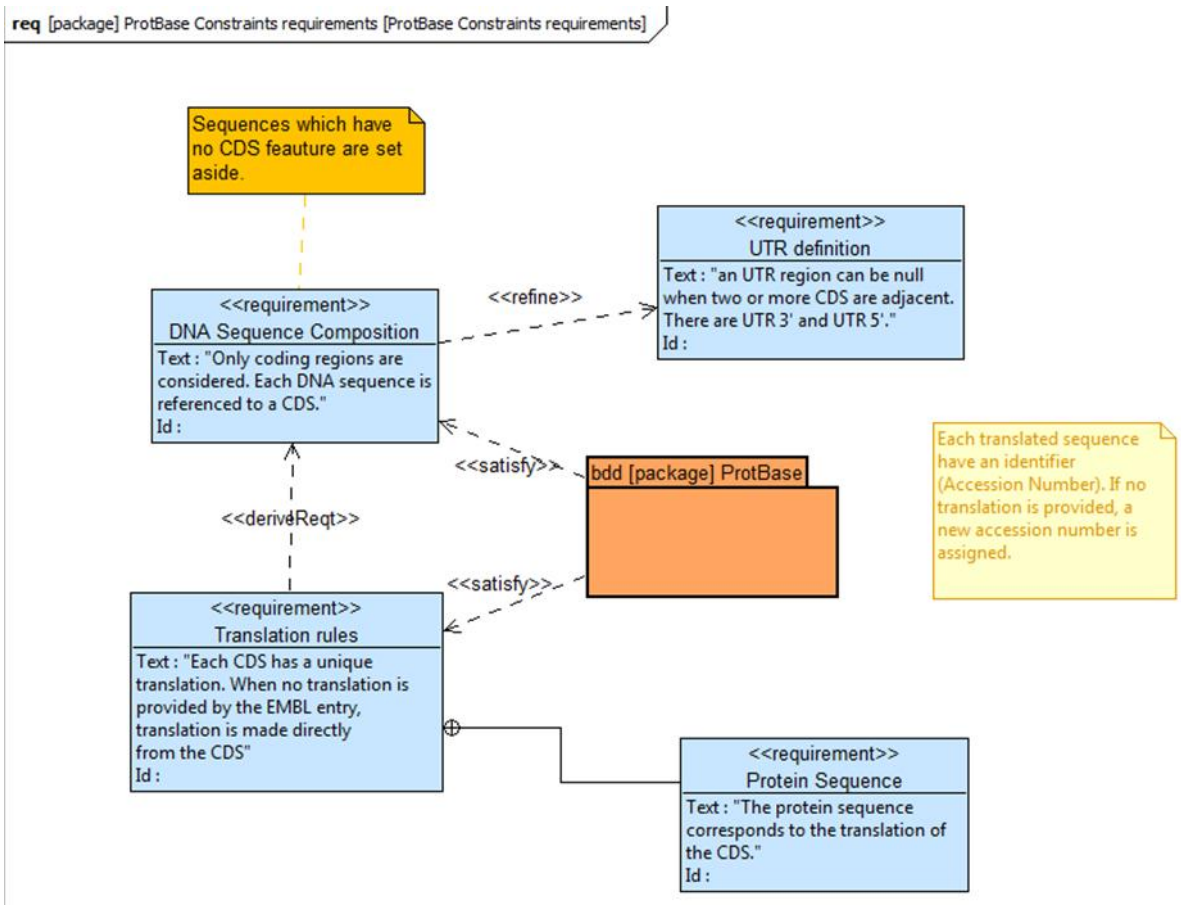


Figure 31: Diagramme des exigences de la base de données.

L'intérêt ici est de proposer à partir des modèles SysML présentés précédemment la base de données relationnelle correspondante. ATL (ATLAS Transformation Language) est un langage permettant de transformer le modèle développé à l'aide de diagramme de classe UML en base de données relationnelle. ATL est disponible en plug-in pour Eclipse[a]. En revanche Rational Rose[b] d'IBM propose une extension d'UML à l'aide des stéréotypes pour modéliser les bases de données relationnelles. Ces deux approches n'ont pas été encore implémentées pour SysML. Un modèle relationnel est proposé ici. Il respecte les besoins (exigences) du modèle ainsi que le diagramme de définition de blocs. La Figure 32 représente le modèle relationnel en question.

[a]Eclipse: http://www.eclipse.org/m2m/atl/atlTransformations/

[b]IBM Data Modeler: http://www-01.ibm.com/software/awdtools/developer/datamodeler/

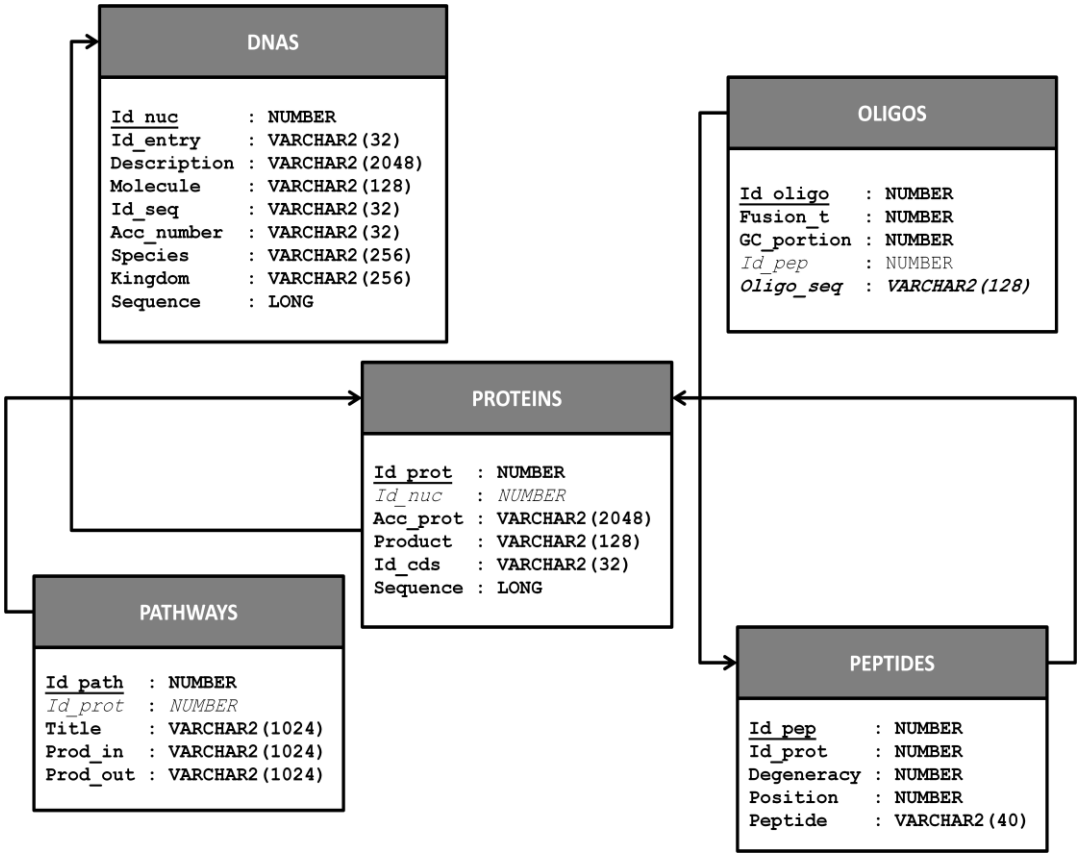


Figure 32: Modèle relationnel de la base de données.

Le code suivant correspond au script SQL permettant la création de la base de données. Il comporte la création des différentes tables et la définition des contraintes d'intégrité.

```
CREATE TABLE dnas
(
  id_nuc INTEGER NOT NULL, -- primary key
  id_entry varchar2(32) NOT NULL, -- id present in id line of embl entry
  description varchar2(2048), -- DE line of embl entry
  molecule varchar2(128), -- molecule type present in the mol type tag of the
source feature in the embl entry
  id_seq varchar2(32) NOT NULL, -- sequence identifier : if cds exists then id_seq
= access_number_cds_rank else id_seq = accession
  acc_number varchar2(32) NOT NULL, -- accession number in AC line of embl entry
  species varchar2(256), -- organism name (species(
  kingdom varchar2(256), -- kingdom in the taxonomy tree present in SP line of the
embl entry
  sequence CLOB NOT NULL, -- sequence : cde if exist else entire sequence of the
embl entry
  CONSTRAINT dnas_pkey PRIMARY KEY (id_nuc)
);

CREATE TABLE proteins
(
  id_prot INTEGER NOT NULL PRIMARY KEY,
  id_nuc INTEGER NOT NULL,
  acc_prot varchar2(32) NOT NULL,
  product varchar2(2048), -- name of the coding sequence in term of enzymatic
activity.
  id_cds varchar2(32) NOT NULL, -- id referencing cds + utr region in the nucleic
base dnas.
  sequence CLOB NOT NULL,
  CONSTRAINT foreign_key FOREIGN KEY(id_nuc) REFERENCES dnas(id_nuc)
);
```

```
CREATE TABLE peptides
(
  id_pep INTEGER NOT NULL PRIMARY KEY,
  degeneracy INTEGER NOT NULL,
  peptide varchar2(40) NOT NULL
);

CREATE TABLE oligos
(
  id_oligo INTEGER NOT NULL PRIMARY KEY,
  fusion INTERGER,
  gc_pour INTERGER,
  oligo_seq varchar2(120),
  CONSTRAINT foreign_key FOREIGN KEY(id_pep) REFERENCES peptides(id_pep)
);

CREATE TABLE pathways
(
  id_path INTEGER NOT NULL PRIMARY KEY,
  id_prot INTEGER NOT NULL,
  title varchar2(1024),
  prod_in varchar2(1024),
  prod_out varchar2(1024),
  CONSTRAINT foreign_key FOREIGN KEY(id_prot) REFERENCES proteins(id_prot)
);
```

2.3. Synthèse

Comme nous l'avons vu dans les chapitres précédents, la dégénérescence du code génétique est l'une des problématiques biologiques à l'origine de la complexité des algorithmes bioinformatiques. Dans le modèle relationnel précédent, les oligopeptides générés à partir de séquences protéiques représentent une signature de ces séquences. Leur traduction inverse en oligonucléotides servira de sondes pour concevoir la biopuce à ADN. Dans la partie suivante, une méthode de traduction inverse complète sera proposée. Une approche IDM pour la conception de cet algorithme sera également détaillée.

3. Traduction inverse d'oligopeptides et IDM

3.1. Définitions et généralités

L'étape qui consiste à produire de l'ARN à partir de l'ADN s'appelle transcription et l'étape qui consiste à produire la séquence protéique correspondante à une séquence ARN est appelée traduction. Ces deux mécanismes biologiques établis après la découverte du code génétique sont naturels. En revanche, lorsqu'on veut passer d'une séquence protéique à une séquence nucléique on parle de traduction inverse ou « backtranslation » ou encore de « reverse translation ». Celle-ci désigne une opération non biologique mais plutôt purement algorithmique. Dans la partie II.1.6, nous avons vu que le code génétique se compose de 4 acides nucléiques et de 20 acides aminés. Chaque acide aminé est codé par un ou plusieurs codons de 3 bases. Parler de traduction inverse de séquences protéiques peut paraître absurde vu la complexité du problème en terme de temps de calcul et en terme d'espace disque lorsqu'il s'agit de travailler sur des séquences protéiques longues. Mais dans le cadre de la conception de sondes à ADN, et lorsqu'on travaille sur des oligonucléotides de petites tailles, le problème devient borné et donc peut être résolu.

Plusieurs travaux se sont intéressés à la traduction inverse de séquences protéiques (Pesole et al., 1988 ; Nash, 1993 ; White et Seffens, 1998 ; Wernersson et Pedersen, 2003 ; Moreira et Maass, 2004 ; Moreira, 2004 ; Suyama et al., 2006 ; Moretti et al., 2006). La plupart d'entre eux se basent sur la traduction inverse d'alignements de séquences protéiques en remplaçant systématiquement les acides aminés par les codons les plus probables en fonction de l'organisme considéré (utilisation préférentielle de certains codons).

L'utilisation d'une telle technique permet de simplifier le passage d'une séquence protéique d'intérêt à la séquence nucléique correspondante. En revanche, l'inconvénient majeur est la perte d'information engendrée par une telle pratique. En effet, le code génétique offre la possibilité de générer autant de séquences que la dégénérescence de la séquence protéique, contrairement au simple remplacement des acides aminés par les codons suivant des critères bien déterminés. Dans l'exemple suivant, nous avons une sortie partielle d'un alignement multiple de séquences protéiques possédant une activité déshydrogénase. Nous remarquons que certaines régions sont conservées et sont signalées par des « * » en dessous de l'alignement. Une des méthodes pour traduire inversement ces séquences conservées en une séquence nucléique est de remplacer les acides aminés par les codons respectifs les plus probables comme c'est le cas pour le programme PAL2NAL (Suyama et al., 2006). La séquence « SEQPROB » obtenue est présentée ci-dessous. Il

est évident que cette séquence est une possibilité parmi d'autres pour écrire la séquence nucléique qui code pour les protéines en question. L'avantage de cette méthode est sa rapidité. Son défaut est le résultat utilisant préférentiellement des codons par rapport à d'autres et donc l'introduction d'un biais considérable surtout si nous travaillons sur des biopuces où toutes le séquences doivent être considérées.

```
Alignement
----------------------------------------
P19871   MTNRLQGKVALVTGGASGVGLEVVKLLLGE
Q9N119   ----MAGWSCLVTGGGGFLGQRIVHLLLEE
P14893   ----MAGWSCLVTGGGGFLGQRIICLLVEE
               *   *****    *     **  *
----------------------------------------
>SEQPROB
NNN NNN NNN NNN ATG GCT GGT TGG TCT TGT TTA GTT ACT GGT GGT GGT GGT TTT TTA
GGT CAA CGT ATT GTT AAA TTA TTA TTA GAA GAA
```

Pour obtenir des résultats complets, il n'est pas envisageable de faire la traduction inverse complète d'une séquence protéique longue car pour la séquence (s = RRRRRRRRRRRR) de longueur 13 aa (acides aminés), on obtient environ 13 milliards de séquences nucléiques correspondantes possibles de longueur 39 mers d'après le code génétique. Les résultats de cette traduction inverse complète demanderont pas moins de 390 Go d'espace disque pour être stockés. Travailler sur des séquences longues (ou la dégénérescence peut atteindre des valeurs très importantes) parait donc non réaliste aujourd'hui. La séquence (t = RRRRRRRRRRRRRRRRRRRRRRRRRR) de longueur 26 aa aura par exemple $1,7\ 10^{20}$ séquences possibles et requiert $11\ 10^{9}$ To d'espace disque, ce qui est inaccessible aujourd'hui. En revanche, travailler sur des séquences courtes – appelées aussi oligopeptides – est possible, lorsqu'il s'agit de concevoir des oligonucléotides pour biopuces à ADN. Dans la suite, un algorithme de traduction inverse complète d'oligopeptides est proposé. Une approche par méta-programmation et une étude de performance puis une parallélisation de l'algorithme seront présentées dans les parties suivantes.

3.2. L'algorithme DegenRev

DegenRev est un algorithme qui a été développé au LIMOS et au LMGE pour réaliser la traduction inverse d'oligopeptides (Missaoui et al., 2007), utile pour la conception de sondes oligonucléotides spécifiques à des protéines. En effet, aucun algorithme dans la littérature ne propose de faire une traduction complète de peptide. Potentiellement, les sondes générées

pourront identifier tous les organismes codant pour cette protéine même si les séquences nucléiques ne sont pas encore répertoriées. Il sera toutefois nécessaire d'identifier les régions les moins complexes permettant de limiter le nombre d'oligonucléotides pour satisfaire la contrainte des formats des biopuces actuelles (2 millions de sondes).

3.2.1. Description de l'algorithme

L'algorithme proposé ici énumère toutes les possibilités de séquences obtenues par traduction inverse d'oligopeptide. Chaque oligopeptide est stocké dans un tableau de longueur « *pl* ». Pour chaque acide aminé « *i* » qui compose l'oligopeptide, la dégénérescence est donnée par « *NC(i)* » : le nombre de codons de l'acide aminé « *i* ». Ensuite, 3 dégénérescences sont calculées :

1. La dégénérescence cumulée donnée par : $d(i) = \prod_{k=0}^{i} NC(k)$
2. La dégénérescence restante donnée par : $r(i) = \prod_{k=i+1}^{pl-1} NC(k)$
3. Et, la dégénérescence passée donnée par : $c(i) = \prod_{k=1}^{i} NC(k-1)\,; c(i) = 1 \; si \; i = 0$

La suite de l'algorithme dépend de l'initialisation de toutes ces variables. En effet, à la fin de l'initialisation, la traduction proprement dite peut commencer. L'algorithme teste pour chaque acide aminé si la valeur « *r(i)* » est atteinte. Si c'est le cas, il passe au codon suivant. Sinon, il répète l'étape de traduction « *c(i)* » fois avec le même codon. En utilisant cette approche l'algorithme réalise *D* boucle avec $D = d(i) \times r(i)$ pour tout i ∈ [0, pl - 1].

3.2.1. Données

Par exemple, pour l'oligopeptide « KIL » composé de 3 acides aminés, les dégénérescences suivantes sont obtenues :

OligoPeptide	*K*	*I*	*L*
I	0	1	2
NC(i)	2	3	6
D	36		
d(i)	2	6	36
r(i)	18	6	1
c(i)	1	2	6

La recherche des oligonucléotides dans la boucle principale de l'algorithme utilise les dégénérescences *d(i)*, *r(i)* et *c(i)* calculées au préalable. Les résultats pour chaque étape sont écrits

dans un fichier de sortie suivant l'ordre d'initialisation de la structure de données contenant les acides aminés et leurs codons. Pour l'exemple précédent, on obtient les résultats de la Figure 33.

Figure 33: Résultats de la traduction inverse de l'oligopeptide « KIL ».

La complexité de l'algorithme est calculée par:

$$Complexité = D \times pl.$$

Où *D* est la dégénérescence de l'oligopeptide et *pl* est sa longueur.

3.2.1. Etude des performances

L'algorithme a été testé sur des données de simulation en faisant varier successivement la longueur de l'oligopeptide et/ou sa dégénérescence. La Figure 34 montre le temps d'exécution de l'algorithme en fonction de la dégénérescence en fixant la longueur de l'oligopeptide. Le temps de calcul reste raisonnable malgré une longueur générée de 54 mers.

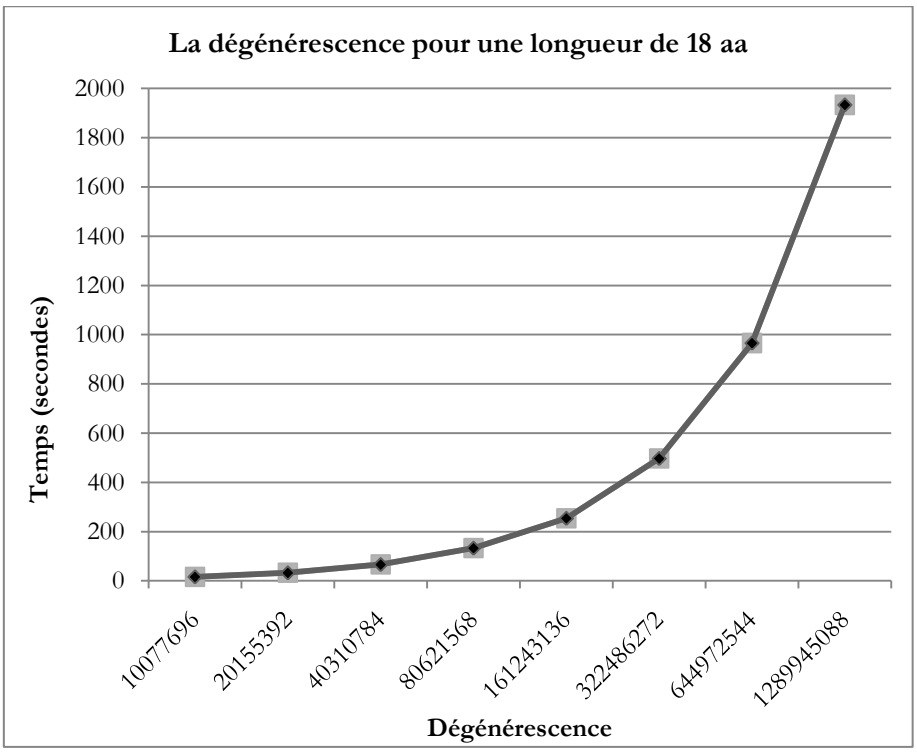


Figure 34: Performance de DegenRev pour une longueur constante.

De plus, l'algorithme montre une variation sensible du temps d'exécution lorsque la dégénérescence de l'oligopeptide est constante en faisant varier la longueur (Figure 35). Cela montre que la longueur influe bien sûr sur les performances de l'algorithme, mais cela montre aussi qu'il est possible d'utiliser le programme pour des données biologiques réelles.

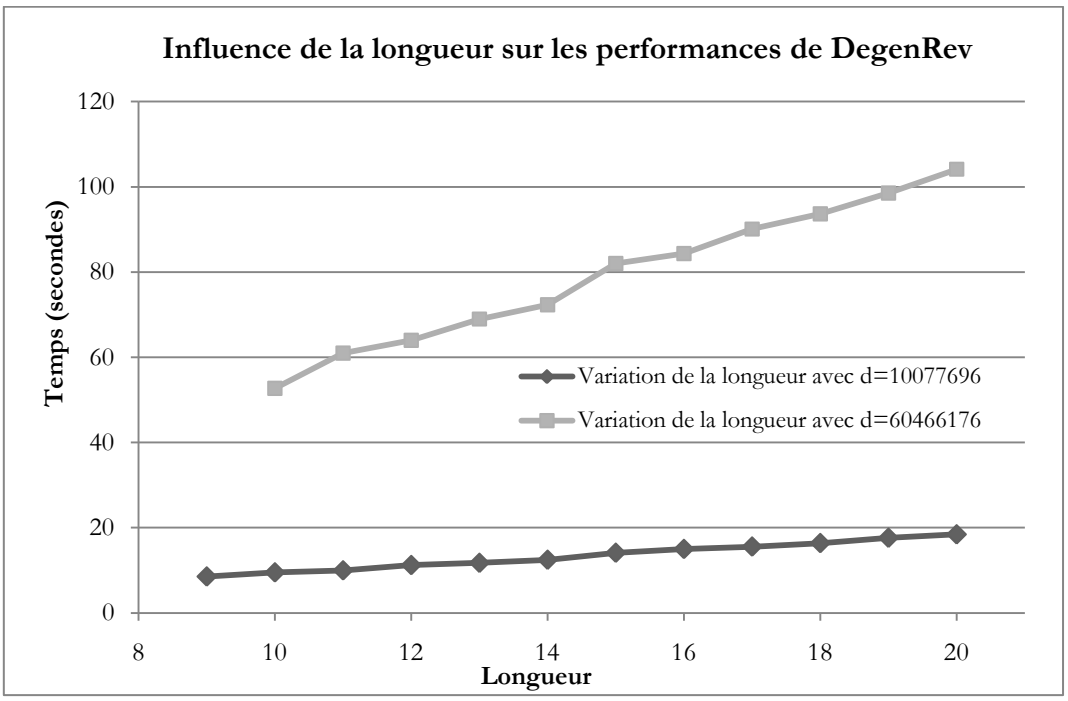


Figure 35: Performance de DegenRev pour une dégénérescence constante.

Pour valider l'algorithme, nous l'avons testé sur les données générées à partir de la bactérie *Pseudomonas* sp. LZT5 ayant un gène qui code pour le naphtalène di-oxygénase. La séquence protéique « Q3LTH2 » intervenant dans la dégradation du naphtalène a été extraite puis utilisée pour générer tous les oligopeptides de 11 acides aminés. Cette taille correspond à une longueur d'oligonucléotides de 33 mers, optimale pour la conception de sondes oligonucléotidiques (Relógio et al., 2002). Nous avons alors obtenu 174 oligopeptides prêts pour une traduction inverse complète. Tous les oligopeptides sont sauvegardés dans un fichier. Pour chaque oligopeptide, un fichier de sortie contenant tous les oligos générés par DegenRev est créé. Les temps d'exécution cumulés sont enregistrés. Les résultats de la Figure 36 montrent que l'exécution totale a pris 101 secondes pour générer tous les oligos possibles. Ce temps est relativement court comparé aux données de simulation. En se basant sur ces résultats, notre algorithme montre des capacités réelles à calculer en un temps relativement court la traduction inverse d'oligopeptides.

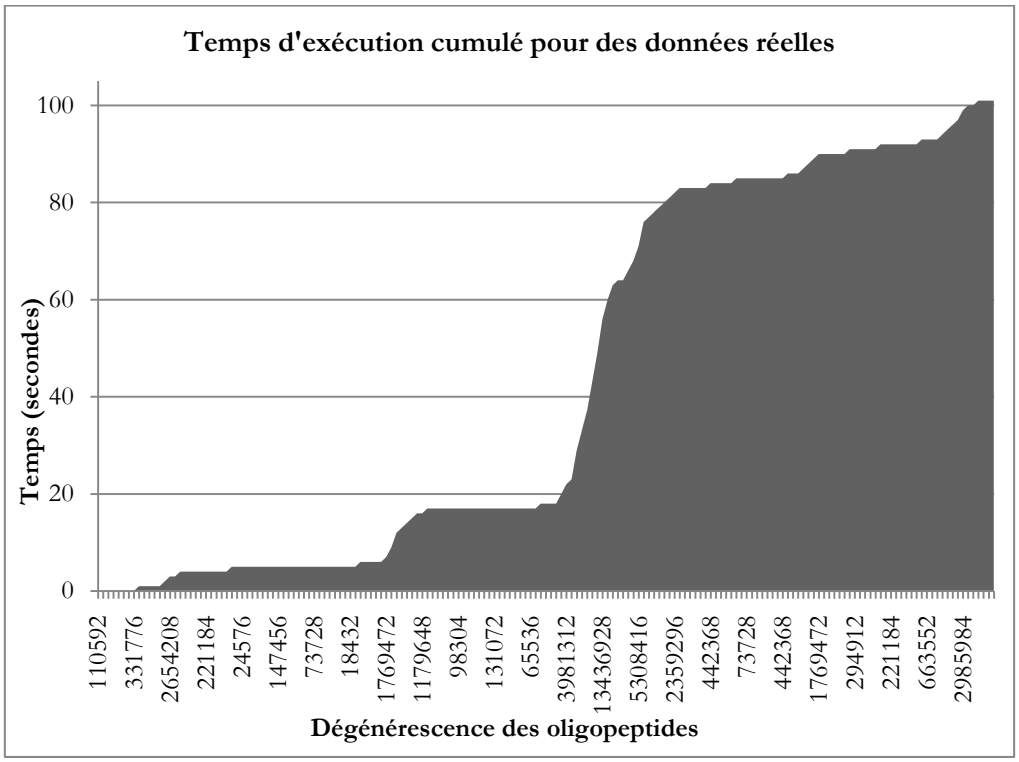


Figure 36: Performance de DegenRev pour des données réelles (174 oligopeptides) de la protéine Q3LTH2.

De plus, sachant que la longueur de l'oligopeptide n'intervient que d'une manière partielle dans le temps global d'exécution, il est donc facile d'appliquer l'algorithme à des longueurs d'oligonucléotides différentes. Enfin, DegenRev est utile pour la conception de sondes spécifiques de voies métaboliques ou d'enzyme d'intérêt pour biopuce à ADN en déterminant tous les oligonucléotides potentiels spécifiques codant pour la protéine impliquée dans la voie métabolique ou l'activité d'intérêt.

3.2.2. StackPRT : un algorithme IDM

Par ailleurs, la traduction inverse d'oligopeptide a été optimisé par une approche de méta-programmation dans le cadre de l'ingénierie dirigée par les modèles en proposant une démarche capable d'absorber au mieux la nature exponentielle du problème de traduction inverse. Cette méthode consiste à écrire dynamiquement le programme qui réalise la traduction inverse pour une longueur donnée. En effet, un algorithme a été développé pour écrire automatiquement à la place du développeur le code source du programme de traduction inverse appelé StackPrt (Missaoui et al., 2008). Par exemple, pour l'acide aminé « NRG », d'une longueur de 3 acides aminés, le code généré correspond au pseudo-code donné par la Figure 37.

La complexité de l'algorithme reste exponentielle, mais, plus l'oligopeptide est long, plus le gain est important par rapport à l'approche classique (DegenRev (*D*×*pl*)). En effet, la complexité est égale à *D*, où *D* est la dégénérescence de l'oligopeptide.

```
Begin
   For i = 1 to NC(1st Amino Acid)
     Push Codon(i) of 1st Amino Acid
     For j = 1 to NC(2nd Amino Acid)
        Push Codon(j) of 2nd Amino Acid
        For k = 1 to NC(3rd Amino Acid)
           Push Codon(k) of 3rd Amino Acid
           Output full stack
           Pop Codon(k) of 3rd Amino Acid
        End For
        Pop Codon(j) of 2nd Amino Acid
     End For
     Pop Codon(i) of 2nd Amino Acid
   End For
End
```

Figure 37: Pseudo-code de l'approche par pile

(Ce code peut être généré automatiquement en fonction de la longueur de l'oligopeptide)

Les avantages de l'approche par génération de code sont :

- Une utilisation limitée de la mémoire (3 fois la longueur de l'oligopeptide + le nombre de boucles sur les indices)
- L'utilisation de la pile limite le nombre d'écriture dans la mémoire. Ceci permet de résoudre une partie de la complexité du problème.
- Le code source est produit automatiquement pour une longueur donnée.

En revanche, les inconvénients d'une telle approche portent sur deux points :

- Chaque algorithme est dédié uniquement à une longueur unique donc une compilation et la génération des exécutables binaires est nécessaire pour chaque longueur.

- Le code permettant de générer automatiquement le code source de l'algorithme est complexe.

Enfin, ces deux algorithmes ont été développés pour obtenir les meilleures performances sur des calculateurs classiques de type micro-ordinateur. Aucune comparaison avec d'autres algorithmes n'a été proposée car à notre connaissance aucun autre algorithme proposant la traduction complète n'a été trouvé dans la littérature.

3.2.3. Comparaison des deux approches de traduction inverse

Contrairement aux tests réalisés dans (Missaoui et al., 2007) pour DegenRev sur une machine Linux (Xeon 2.4 GHz, 2 Go RAM), les tests de performances pour cette comparaison on été effectués sur une machine Linux multiprocesseurs 8 × AMD Opteron Core duo 1.8 GHz. Les deux algorithmes sont développés en langage C et ont été compilé avec la version 3.4.6 de GCC en utilisant l'option LFS (Large File System). DegenRev est très performant pour les oligopeptides ayant une dégénérescence inférieure à 10^7. Il est capable de calculer – dans un temps raisonnable – les traductions inverses de tous les oligopeptides ayant une longueur comprise dans [1 … 14]. En pratique, pour la conception de sondes ADN, les longueurs les plus intéressantes sont dans [8 … 17] (équivalentes à [24 … 51] mers). Dans le Tableau 11 les temps d'exécution pour les longueurs [8 … 14] sont donnés pour les deux algorithmes. Les oligopeptides testés sont : « `TRALA ... LA` » avec NC(T) = 4, NC(R)= 6, NC(L)=6 et NC(A)=4. Comme prévu, StackPRT est plus performant que DegenRev. Le Tableau 11 donne les ratios de performances entre DegenRev et StackPRT. La Figure 38 donne le temps d'exécution des deux algorithmes.

Tableau 11: Temps d'exécution en secondes pour les deux algorithmes de traduction inverse (DegenRev et StackPRT).

Nombre d'acides aminés	Temps d'exécution StackPRT	Temps d'exécution DegenRev	Nombre d'oligonucléotides générés	Ratio DegenRev / StackPRT
8	0,066	0,161	331,776	2.44
9	0,281	0,725	1,327,104	2.58
10	1,631	4,497	7,962,624	2.76
11	6,932	19,571	31,850,496	2.82
12	41,845	130,968	191,102,976	3.13
13	163,773	566,661	764,411,904	3.46
14	996,896	26042,736	4,586,471,424	26.12
15	4119,544	N.C	18,345,885,696	N.C

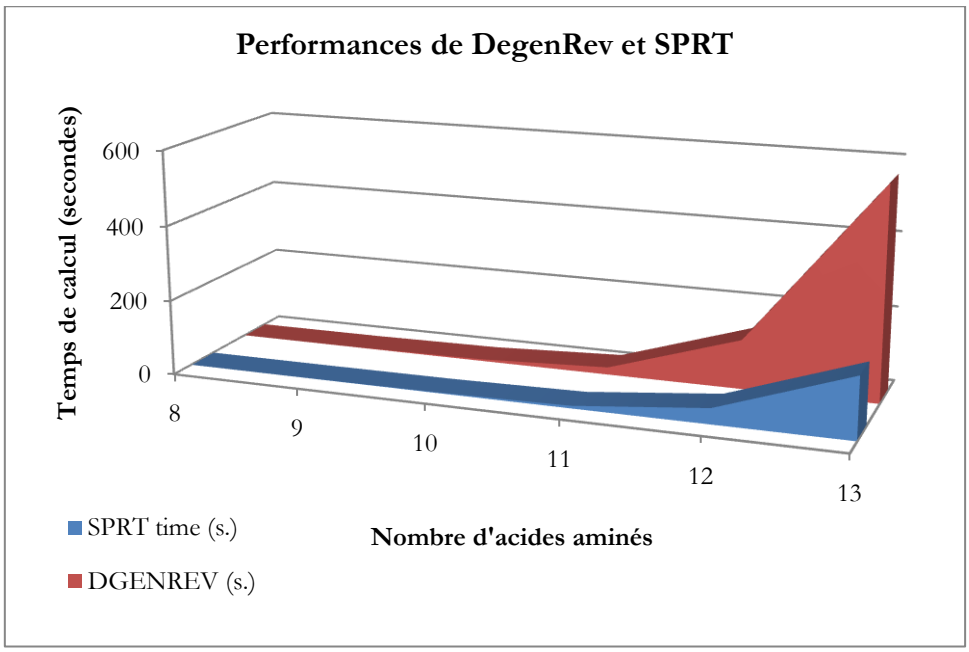


Figure 38: Performance des deux algorithmes.

Nous avons constaté que le problème de stockage des résultats de traduction inverse se pose lorsque la dégénérescence dépasse un certain seuil et il devient impossible d'envisager de stocker les données sur des disques durs. L'oligopeptide « `TRALALALALALALALA` » par exemple, produit 440 milliards de séquences qui ont besoin de 22 téraoctets pour être stockées. Une solution serait de compresser les fichiers de résultat en utilisant les commandes classiques Linux telles que *compress* ou *gzip* ayant un facteur de compression de 15.

Pour les oligonucléotides d'une longueur comprise entre 24 et 50 mers, les facteurs de compression sont donnés par la courbe de la Figure 39. Plus la taille est importante, plus le gain est conséquent. Enfin, pour la comparaison des deux algorithmes sur des données réelles, nous avons repris la séquence Q3LTH2 utilisé précédemment. Les résultats sont donnés par la Figure 40. Ces deux algorithmes sont performants sur des données simulées ainsi que sur des données biologiques réelles.

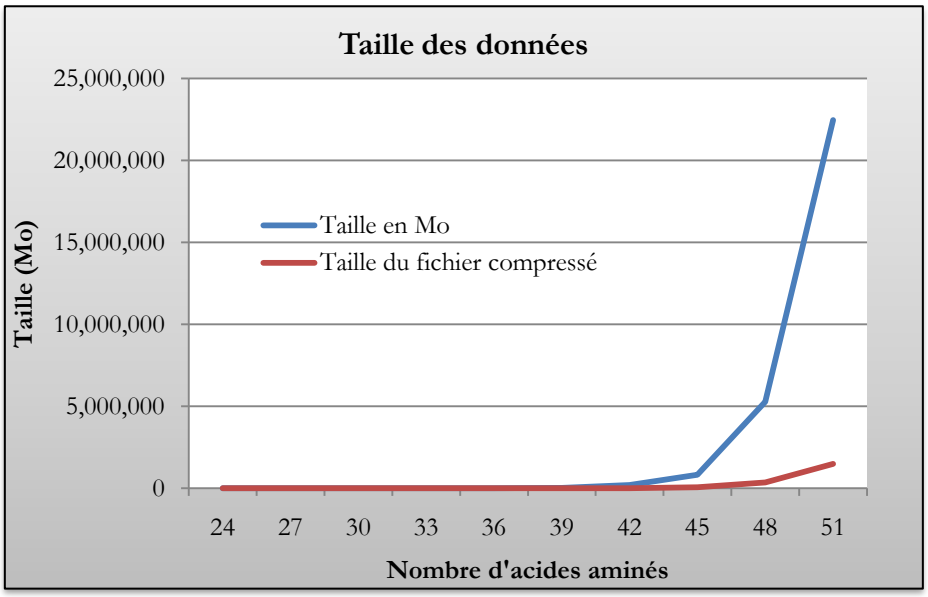


Figure 39: Gain d'espace disque par compression pour les longueurs [24 … 51] en acides aminés.

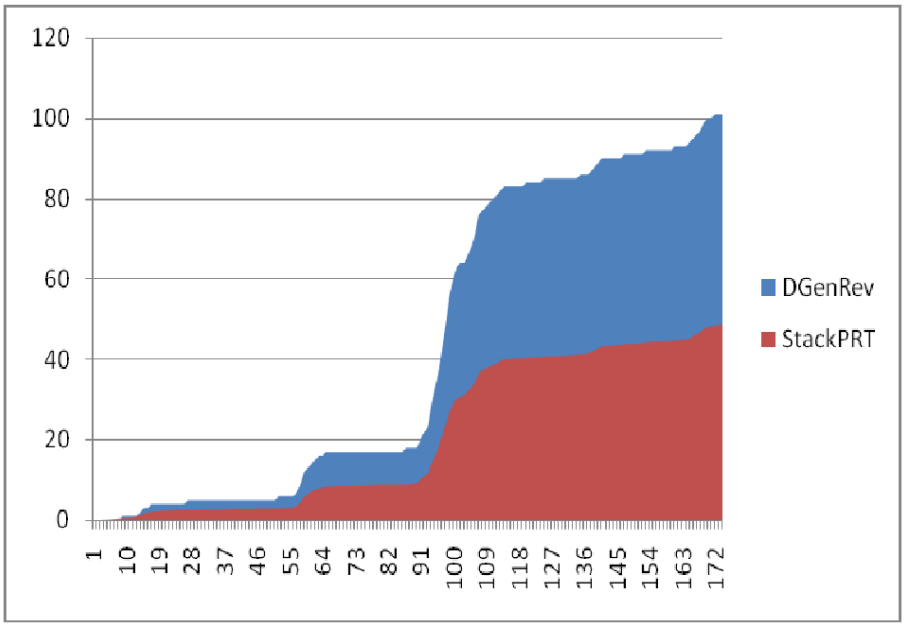


Figure 40: Comparaison du temps d'exécution cumulé en secondes des deux algorithmes pour 172 séquences.

3.2.4. Amélioration des performances par parallélisation

Nous proposons également une approche IDM pour l'utilisation de plateformes parallèles telles que les SMP ou les clusters. Nous la présenterons dans la partie suivante. La génération de code effectuée correspond à une transformation de modèle au sens IDM. L'idée est de proposer un programme permettant d'écrire le code source suivant la traduction inverse d'oligopeptide envisagée. En effet, une telle approche s'inscrit dans l'optimisation de la traduction inverse

d'oligopeptides. Elle permet de fournir le code source du programme pour une plateforme donnée et pour une longueur donné en se basant sur l'algorithme StackPRT décrit plus haut. Le modèle de l'architecture est décrit par le diagramme de classe UML de la Figure 41. Les classes « Programme Oligo », « Programme Segmentation » et « Programme Fichier » héritent de la classe « Programme C » et correspondent aux classes : classe de traduction inverse pour un seul oligopeptide, classe de la segmentation d'un fichier et classe de la traduction d'un fichier complet d'oligopeptides respectivement. Les scripts Shell peuvent être générés pour un cluster ou une machine SMP. Toutes ces classes décrivent des modèles de programmation ou d'architecture matérielle. Ce sont des méta-classes au sens IDM.

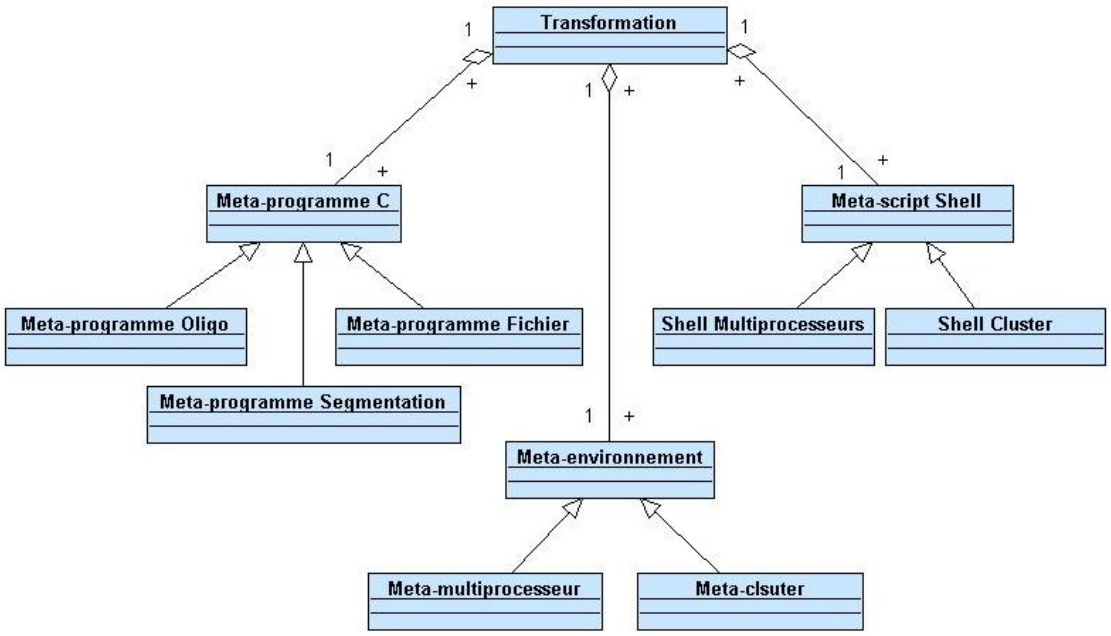


Figure 41: Méta-modèle UML pour sa transformation par génération de code.

L'intérêt d'une telle approche est de pouvoir paralléliser le calcul des traductions inverses de plusieurs oligopeptides contenus dans un fichier, provenant d'une séquence d'intérêt. Selon l'environnement de calcul choisi par l'utilisateur, la transformation par génération de code concerne les codes de segmentation et du programme de traduction spécifique pour un fichier. L'utilisateur choisit alors la stratégie de parallélisation consistant à répartir les calculs sur le nombre de processeurs. La répartition se fait en équilibrant la charge de calcul à partir de la dégénérescence moyenne de tous les oligopeptides et répartit le calcul pour chaque fragment sur plusieurs processeurs. Les fragments contiennent des oligopeptides de telle façon que la dégénérescence de chaque fragment soit la plus proche possible de la dégénérescence moyenne.

La Figure 42 représente le code permettant d'estimer la dégénérescence moyenne par fragment. Cette dégénérescence est recalculée si le seuil est dépassé lors de la création des fragments.

```
void estimDgenParFichier(FILE * inOligoFile, int nbProc)
{
         int   i = 0,
               j = 0;

         for(i = 0; i < nbProc ; i++)
         {
            tNbDGenParFichier[i] = 0;
         }

         i = 0;
         fgets(buffer,255,inOligoFile);
         while (!feof(inOligoFile))
         {
            tNbDGenParOligo[i] = 1;
            for(j = 0; j <= 10 ; j++)
            {
               tNbDGenParOligo[i] *=  TabCodage[buffer[j]].nbCodons;
            }
            dGenTotale += tNbDGenParOligo[i];
            i++;
            fgets(buffer,255,inOligoFile);
         }
         dGenSeuilParProc = dGenTotale / nbProc;

         rewind(inOligoFile);
}
```

Figure 42: Code d'identification de la dégénérescence moyenne pour répartir la charge de calcul.

La parallélisation du calcul sur un multiprocesseur utilise plusieurs CPUs qui correspondent au nombre de fragments du fichier d'entrée. L'exemple réel utilisé dans la partie III.3.2.3 est réutilisé pour être exécuté en employant cette méthode d'équilibrage de charge par dégénérescence et permet d'obtenir les résultats de laFigure 43.

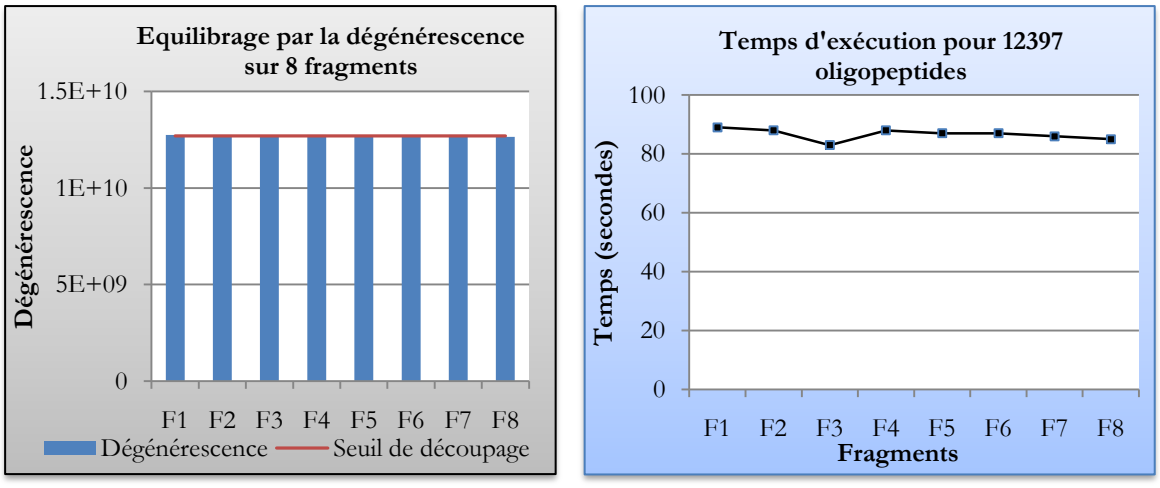


Figure 43: (a) Résultats après équilibrage de 8 fragments sur une machine multiprocesseurs (8 Core-Duo 1.80Ghz, 128 Go de RAM) pour 174 oligopeptides. (b) Temps d'exécution pour 12397 oligonucléotides

De plus, l'augmentation du nombre de CPUs utilisées permet de diminuer le temps de calcul global comme le montre la Figure 44. En effet, l'indépendance entre les différents jobs permet une parallélisation maximale du calcul grâce à un équilibrage de charge; ainsi le caractère exponentiel de l'algorithme est fortement absorbé.

Figure 44: Temps de calcul en fonction du nombre de cœurs utilisés.

3.3. Synthèse

Dans le cadre de la conception de sondes ADN spécifiques à des protéines d'intérêt, la technique de traduction inverse complète d'oligopeptide est utile pour obtenir toutes les combinaisons d'oligonucléotides susceptibles de le coder et ainsi avoir une approche exploratoire pour identifier de nouveaux micro-organismes. Une telle technique combinée à une approche de génération de code dans un cadre d'ingénierie par les modèles réduit la complexité due au code génétique. Seules les meilleures sondes (spécifiques) seront gardées. En revanche, la quantité de données à tester (par une comparaison BLAST contre les banques de données génomiques) reste très importante et requiert l'utilisation de puissance de calcul plus importante que ce qui est disponible sur un multiprocesseur ou sur un cluster. Une adaptation du programme BLAST à la grille EGEE de calcul sera présentée dans la partie suivante.

4. Adaptation du programme BLAST à la grille de calcul EGEE

4.1. Présentation

Comme nous l'avons vu précédemment, le programme BLAST est largement utilisé dans les applications de conception de sondes pour puces à ADN. Il permet d'identifier les hybridations croisées potentielles d'un oligonucléotide par rapport à une banque de données choisie. Son utilisation à grande échelle pour trouver les similarités d'un grand nombre d'oligonucléotides permettra de diminuer le temps de calcul global de tous les oligonucléotides générés – relativement nombreux – et permettra ainsi de concevoir des sondes dans un temps raisonnable. En effet, le nombre d'oligonucléotides générés pour une seule séquence protéique d'intérêt est de l'ordre d'une dizaine voir de centaines de millions d'oligonucléotides qui devront être comparés à la base de données *ProtBase* présentée dans la partie III.2. Cette étape peut prendre sur une seule machine quelques jours à quelques semaines pour une seule séquence. Pour plusieurs séquences cela prendra quelques mois à quelques années. Plus de détails seront apportés dans la suite de cette section.

4.2. Description de la grille EGEE

La grille EGEE (Enabling Grids for E-SciencE) est la plus grande infrastructure de grille de calcul institutionnelle dans le monde. Elle compte plus de 120 organisations virtuelles (VO), contient 250 sites à travers 50 pays et offre près de 80.000 CPUs et 20.000 To d'espace de stockage avec un transfert massif de données à plus de 1.5 Go/s. EGEE est gérée comme étant une collection de plusieurs VOs. Chaque VO est un ensemble d'utilisateurs de la grille répartis suivant leur centre d'intérêt et leur besoin et pouvant collaborer avec d'autres membres du groupe en partageant des ressources (données, logiciels, expertise, CPU, stockage…) indépendamment de leur situation géographique. EGEE compte aujourd'hui 126 VOs concernant différentes applications telles que l'astrophysique, la physiques des hautes énergies, la biomédecine, la bioinformatique et l'informatique appliquée à la chimie. Chaque personne physique doit être inscrite à une VO d'EGEE pour bénéficier de l'utilisation des ressources. La VO Auvergrid est la VO de la région de l'Auvergne qui fait partie de la grille EGEE. Elle compte environ 1200 CPUs et 240 To d'espace disque répartis sur plusieurs sites. C'est une VO appropriée pour exécuter un BLAST parallèle. En effet, elle compte suffisamment de CPUs et

suffisamment d'espace de stockage pour lancer un BLAST parallèle à grande échelle sachant que la base de données ProtBase a une taille de quelques gigas octets.

4.3. Architecture et matériels

Pour accéder à la grille EGEE, il faut s'inscrire d'abord à une VO, en l'occurrence ici Auvergrid. Un certificat est alors attribué à l'utilisateur et lui permet d'accéder aux ressources disponibles dans cette VO. Après avoir installé le certificat sur la machine servant à exploiter les ressources de la grille, appelée UI (User Interface) – ici un biprocesseur 1.8 GHz ; 1Go de RAM –, l'utilisateur crée alors son proxy lui permettant de s'authentifier et d'acquérir les autorisations nécessaires pour utiliser la grille. Le middleware utilisé pour la gestion des jobs sur la grille à partir de l'UI est gLite. Enfin, la grille EGEE, et en particulier la VO Auvergrid, est basée sur GT (Globus Toolkit). L'architecture globale utilisée est résumée dans la Figure 45. Chaque job soumis utilisera une CPU au niveau d'un WN (Worker Node) faisant partie d'un CE (Computing Element).

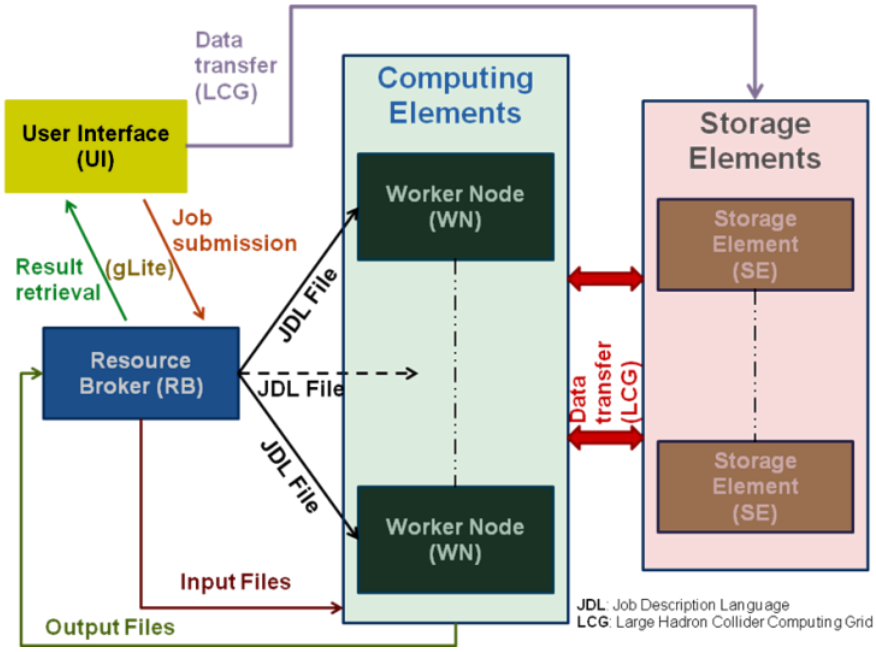


Figure 45: Architecture globale d'une grille de calcul EGEE.

Dans le contexte de conception d'oligonucléotides pour les biopuces à ADN, la recherche de similarité, comme nous l'avons vu précédemment, est la tâche la plus significative en terme de temps de calcul dans le processus. En effet, un grand nombre de BLAST doit être effectué sur les oligonucléotides pour identifier ceux qui présentent de fortes similarités et donc des risques d'hybridations croisées. L'idée est donc de générer à partir d'une protéine d'intérêt tous les oligopeptides d'une longueur donnée, et ensuite de générer par traduction inverse pour chaque oligopeptide tous les oligonucléotides potentiels. La base de données contre laquelle sera effectué le BLAST comporte tous les transcriptomes (CDS + UTR 5' + UTR 3') des microorganismes

(champignons FUN + procaryotes PRO + environnement ENV). Cela permettra de concevoir des biopuces fonctionnelles pour les activités métaboliques d'intérêt ciblées. Dans la suite, le programme permettant de paralléliser l'algorithme BLAST sur la grille EGEE sera détaillé et ses performances seront démontrées.

4.4. BLASTOG : parallélisation du BLAST sur la grille EGEE

4.4.1. Description générale

Un nombre important de BLAST élémentaires est nécessaire pour la recherche des similarités pour la sélection des meilleures sondes pour biopuces ADN. Par exemple, pour la séquence Q69GM4 (Vinyl chloride reductase precursor), 514 oligopeptides de 6 acides aminés sont générés. Le BLAST des 653 696 oligonucléotides de 18 mers générées à partir de ces oligopeptides contre la base de données de 3.2 Go contenant environ 2,5 millions de séquences, met approximativement 2 semaines sur un seul processeur et 3 jours sur une machine SMP ou un cluster de 30 CPUs. Le challenge est de réduire significativement le temps de calcul global en déployant le calcul sur la grille EGEE. Une parallélisation du BLAST appelée BLASTOG (Missaoui et al., 2009) a été développée au LIMOS et au LMGE et permet de déployer à grande échelle un BLAST sur cette grille. La solution proposée est basée sur la technique de fragmentation du fichier d'entrée – contenant les oligonucléotides à comparer – en plusieurs parties équivalentes pour ensuite lancer les BLASTs élémentaires sur ces fragments.

4.4.1. Démarche de parallélisation du BLAST

La démarche de parallélisation du BLAST sur la grille EGEE proposée ici peut être décomposée en plusieurs étapes (voir la Figure 46):

a- **Sélection de la base de données** : la base de données sélectionnée à l'aide de l'outil BLASTOG est formatée, compressée, copiée puis enregistrée au niveau des Storage Elements (SE). La base de données par défaut contient tous les CDS (procaryotes, champignons et environnement) extraits de la banque EMBL. Par ailleurs, l'utilisateur peut choisir sa propre base de données.

b- **Fragmentation du fichier d'entrée** : le fichier contenant les oligonucléotides à comparer est fragmenté en plusieurs sous-fichiers. Le nombre de fragments est un paramètre utilisateur. Les jobs correspondants aux fragments sont ensuite soumis simultanément à la grille EGEE en lançant le BLAST sur les différents fragments.

c- **Génération des scripts Shell et des fichiers JDL** : pour chaque fragment créé, un script Shell nécessaire à l'exécution du job correspondant sur la grille est généré. Un

fichier JDL permettant de décrire le job est aussi généré. Le script ainsi que le fragment seront copiés sur les CE lors de la soumission du JDL.

d- **Soumission et supervision des jobs** : chaque fichier JDL correspond à un job soumis à la grille EGEE. Chaque soumission est contrôlée par l'outil : si la soumission échoue, BLASTOG resoumet le job. De plus, lors de l'exécution du job, l'outil utilise les commandes gLite pour superviser son bon déroulement (Done(Success)). Si le job échoue (Failed, Aborted…), le job est resoumis jusqu'à ce qu'il se termine normalement. Lorsque le job se termine avec succès, il doit attendre que tous les autres jobs soient également terminés pour passer à la dernière étape.

e- **Récupération et concaténation des résultats** : lorsqu'un job est terminé, son résultat est récupéré sur l'UI en attendant que tous les résultats de tous les jobs soient terminés pour procéder à la concaténation. Les résultats sont concaténés dans le même ordre que le fichier d'entrée.

La boucle de supervision des jobs est donnée par le code suivant qui récupère en temps réel les statuts des différents jobs. Ceci étant réalisé en utilisant l'API[1] gLite. Lorsque le job échoue (Failed) ou est avorté par le système (Aborted) ou se termine inopinément avec des erreurs, il est resoumis à la grille immédiatement sachant qu'il est resoumis automatiquement 3 fois d'après les spécifications par défaut du JDL[2].

```
#Récupération du status du job
$status = j_status($count);
#Test sur le statut du job
if ($status eq "Aborted" || $status eq "Done(Failed)"
                         || $status eq "Done(Exit Code !=0)" )
{
        system ("rm -f $input.$count.job_id");

        #Soumission du job
        system ("glite-wms-job-submit
                 -a
                 -r cclcgceli03.in2p3.fr:2119/jobmanager-bqs-short
                 -o $input.$count.job_id  $input.$count.jdl
                 > $input.$count.submit");
        open(SUBMISSION,"$input.$count.submit");

        #Vérification des érreurs lors de la soumission
        while (my $lines = <SUBMISSION>)
        {
             if ($lines =~ /[\w|\W]*Error[\w|\W]*/)
             {
                   system ("rm -f $input.$count.job_id");
                   redo REJOBS;
             }
        }
}
```

[1]API: Application Programming Interface
[2]JDL: Job Description Language

Afin d'assurer le transfert des données, en particulier la base de données de BLAST, la taille des données transférées pour chaque job est contrôlée lors de l'exécution. Si la taille transférée ne correspond pas à la taille réelle de la base, le transfert est à nouveau effectué. Parfois des incidents au niveau du réseau peuvent survenir au sein de la grille comme par exemple la panne d'un CE ou la coupure du réseau entre deux ou plusieurs machines. Pour passer outre, une réplication de la base ou un contrôle des données transférées est effectué.

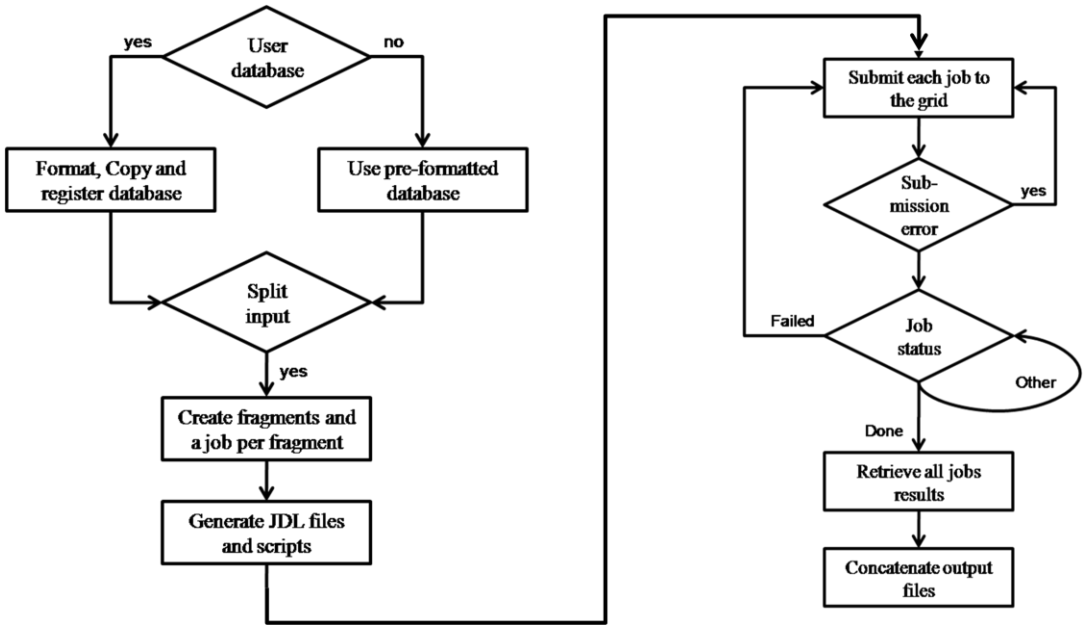


Figure 46: Workflow de la parallélisation du BLAST sur grille de calcul.

Les jobs sont indépendants mais la concaténation se fait uniquement lorsque tous les résultats sont disponibles. (Missaoui et al., 2009)

En ce qui concerne la soumission du job, le WMS peut ne pas trouver de chemin valide vers les ressources, ce qui peut le faire échouer. BLASTOG permet de resoumettre le job immédiatement.

4.4.2. Etude des performances

4.4.2.1. Etude à « petite échelle » – 3000 oligonucléotides

Pour les tests de performance de BLASTOG, une base de données de 3.2 Go a été considérée. Pour les séquences oligonucléotides, un algorithme de génération aléatoire de séquences nucléiques a été développé afin de préserver le maximum d'exhaustivité au niveau de la recherche de similarité et pour produire à chaque test des données différentes. Tout d'abord, un BLAST séquentiel a été testé (voir la Figure 47). Les paramètres retenus pour l'exécuter sont ceux utilisés pour détecter les hybridations croisées dans le cadre de la conception d'oligonucléotides pour biopuces à ADN ; à savoir une e-value [1] égale à 1000, une recherche sur le brin + [2], une taille

du mot égale à 7 et le nombre de séquences de la base, ayant leur alignement affiché, est égal à 500. Tous les autres paramètres étant ceux par défaut. Pour la simulation, plusieurs fichiers (500, 1000, 2000 et 3000 séquences) contenant des oligonucléotides générés aléatoirement ont été créés. La machine sur laquelle a été exécuté le BLAST est équipée d'un Opteron 1.8 GHz, avec 128Go de RAM. Le programme présente un comportement linéaire sur un seul processeur, ce qui était prévisible. 3 heures et 17 minutes ont été nécessaires pour le traitement de 1000 séquences, 6 heures et 40 minutes pour traiter 2000 séquences et 9 heures et 48 minutes pour 3000 séquences.

Figure 47: Temps d'exécution du BLAST sur une seule CPU.

Le comportement du BLAST est quasi-linéaire en fonction du nombre de séquences de même longueur.

Dans un deuxième temps, une série de simulations ont été lancées sur la grille EGEE à l'aide de BLASTOG. Dans ce contexte de calcul distribué, il est nécessaire de considérer des temps supplémentaires qui s'additionne au temps d'exécution réel sur les WNs [3], comme par exemple le temps de formatage, de copie et d'enregistrement de la base de données, le temps de fragmentation et de création des fichiers nécessaires à la distribution et le temps de soumission et de récupération des données. En général ces temps de calcul représentent un pourcentage réduit par rapport au temps d'exécution global sur une grille de calcul. En effet, le temps de soumission moyen par exemple est de 6 secondes. Pour 20 fragments on obtient alors 180 secondes = 3

[1]E-value: paramètre BLAST qui détermine le score des segments d'alignements
[2]Brin +: C'est le brin d'ADN codant
[3]WNs: Worker Nodes

minutes de temps de soumission pour tous les fragments ce qui représente par exemple environ 4% du temps global pour 3000 séquences (Figure 48).

De plus, d'après les résultats, lorsqu'on augmente le nombre de CPUs, le temps de calcul diminue, surtout quand on travaille sur des fichiers de taille importante. En effet, le temps de calcul pour un fichier de 3000 séquences passe de 2h sur 10 CPUs à 1h et 22 minutes sur 20 CPUs, soit un gain de 33.2%. D'autre part, BLASTOG est capable de calculer la similarité de 3000 oligonucléotides 7 fois plus rapidement sur 20 CPUs de la grille que sur une seule CPU (Figure 49).

Figure 48: Performance de BLASTOG pour 3000 séquences sur un nombre croissant de CPUs.

La diminution du temps de calcul est plus importante lorsqu'on augmente le nombre de CPUs et lorsqu'on travaille sur un grand nombre de séquences

Nous remarquons aussi que le speed-up augmente lorsque le travail est réalisé sur des données importantes et lorsque le nombre de fragments augmente.

Figure 49: Speed-up de BLASTOG sur Grille de calcul à petite échelle.

Le speed-up est croissant en fonction du nombre de CPUs surtout pour des données importantes.

Par ailleurs, quand BLASTOG est exécuté sur un fichier de 3000 oligonucléotides pour 30 CPUs, le speed-up augmente jusqu'à 8. Le comportement du logiciel devient un peu plus intéressant quand il est exécuté sur 4000 oligonucléotides en utilisant 50 CPUs où le speed-up atteint 13,5.

4.4.2.2. Etude à plus « grande échelle » – 20.000 oligonucléotides

Pour la deuxième série de test, nous avons considéré un fichier d'entrée de 20.000 séquences d'oligonucléotides. L'exécution de ce fichier d'entrée contre la même base de référence de BLAST et sur la même machine (cf. partie 4.4.2.1) a pris 2 jours et 19 heures environ. Sachant que la VO Biomed compte 20.000 CPUs, nous avons lancé un calcul à grande échelle mais avec des jobs s'exécutant rapidement afin de tester le comportement de BLASTOG et de la grille (Tableau 12). En fait, chaque job est estimé à 4 minutes environ mais nous en avons lancé un millier donc chaque fichier contient 20 séquences oligonucléotidiques. Les résultats donnés par le tableau suivant sont ceux attendus.

Tableau 12: Résultat de la simulation pour 1000 petits jobs.

Temps (mn)	0	108	203	1354	1500	2876	3060
Done	0	164	358	598	617	657	662
Other	1000	120	2	0	0	0	0
Running	0	434	423	294	266	237	232
Scheduled	0	282	217	108	117	106	106

En effet, les performances données par la Figure 50 montrent que ce plan d'exécution représente l'inconvénient de solliciter la grille pour des jobs très courts. Cela a comme incidence directe de voir le nombre de jobs (Done) parmi les 1000 soumis croitre rapidement jusqu'à atteindre 600 jobs après 1500 minutes puis ralentir considérablement pour atteindre seulement 660 jobs après 3000 minutes. Ces résultats montrent que la soumission d'un grand nombre de jobs courts à la grille EGEE, ne permet pas l'obtention de tous les alignements au bout de la durée théorique d'exécution sur une seule machine (dans l'exemple, 2 jours et 19 heures). Cette simulation ne donne pas de bons résultats pour le calcul de l'alignement en question car nous n'avons pas obtenu le résultat final escompté. Le problème de fragmentation sur grille de calcul nécessite de connaitre le comportement de chaque fragment mais également d'être conscients des délais de soumission sur une grille.

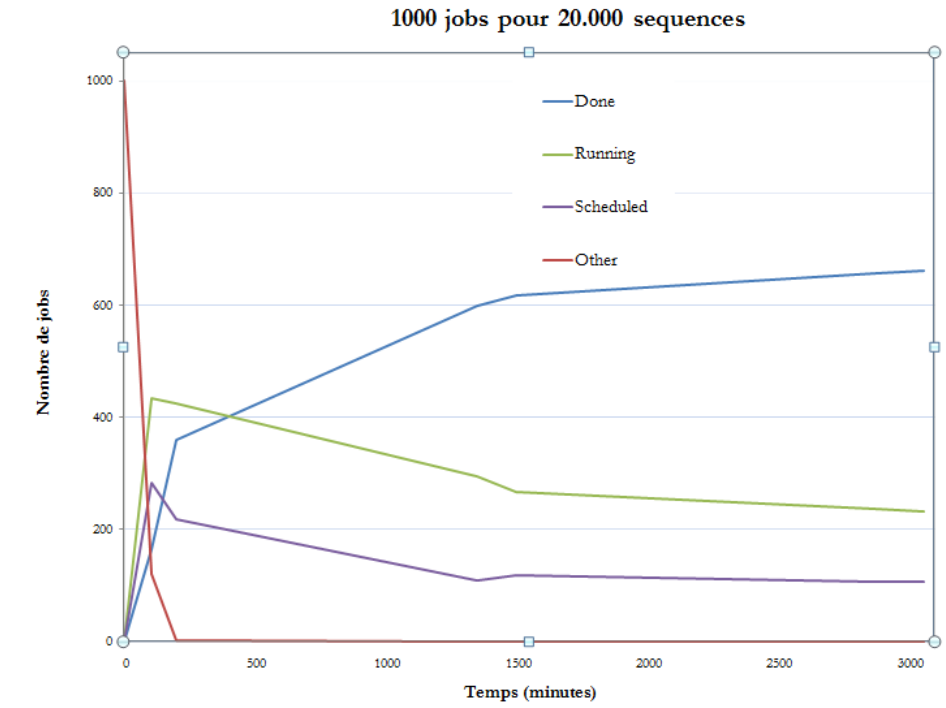


Figure 50: Performances de BLASTOG sur la Grille EGEE pour 1000 jobs courts.

Notre algorithme permet de superviser l'évolution de chaque job correspondant à une partie du calcul. Chaque fragment doit donner son résultat d'alignement BLAST, faute de quoi, le résultat global est erroné car des alignements manquent. Il est donc primordial de récupérer les

résultats de chaque alignement. En effet, tant que le résultat d'un fragment n'est pas disponible, l'alignement final n'est pas fini. L'état de chaque job donne une idée de la disponibilité des résultats des fragments, mais il est obligatoire de vérifier son existence à la fin du calcul sur la grille. Dans la simulation de la Figure 50, seuls les jobs terminés avec des résultats respectifs disponibles sont considérés. Enfin, environ 2 jobs sur 3 sont resoumis à la grille après 12 heures de calcul pour notre exemple.

Afin de trouver un meilleur compromis pour l'obtention des résultats d'alignement, un test de 10 jobs longs (environ 6 heures et 40 minutes pour chaque job sur la grille) a été lancé. Dans cet exemple, nous avons lancé 10 fois la même simulation à la grille EGEE. Les calculs sont lancés l'un après l'autre en respectant la même charge de calcul. Les résultats sont donnés par le Tableau 13. Les résultats des jobs sont mesurés toutes les 12 heures, jusqu'à 72 heures. Les simulations qui ont rencontré des problèmes d'exécution sont surlignées en rose.

Tableau 13: Résultats des simulations pour 10 jobs sur la grille EGEE.

Temps H	0	12	24	36	48	60	72
Simu 1	0	3	4	7	7	10	10
Simu 2	0	0	0	0	0	0	0
Simu 3	0	0	0	0	0	0	0
Simu 4	0	0	0	0	0	1	1
Simu 5	**Erreur**	**NC**	**NC**	**NC**	**NC**	**NC**	**NC**
Simu 6	**Erreur**	**NC**	**NC**	**NC**	**NC**	**NC**	**NC**
Simu 7	0	0	3	8	10	10	10
Simu 8	0	4	10	10	10	10	10
Simu 9	0	6	10	10	10	10	10
Simu 10	0	0	10	10	10	10	10

La simulation (terminée) la plus rapide a mis 15 heures et 38 minutes pour s'exécuter tandis que la simulation la plus longue a mis 53 heures et 27 minutes pour s'exécuter, sur les 10 simulations que nous avons réalisées. Cinq simulations n'ont donné aucun résultat, et deux simulations se sont terminées inopinément avec des erreurs, deux autres simulations ont continué à ne donner aucun résultat au bout de 72 heures et une simulation n'ayant qu'un seul job de terminé au bout de 63 heures environ. Dans le cas où les simulations se sont terminés brusquement, nous sommes remontés à l'origine de l'erreur et nous avons remarqué qu'il s'agissait souvent d'un problème de connectivité réseau ou d'un problème d'arrêt de serveurs distants (élément de la grille : WMS, CE) comme par exemple l'erreur suivante survenue lors de la récupération du statut d'un job:

```
Error while calling the "Status:getStatus" native api
Unable to retrieve the status for:
https://wms.grid.sara.nl:9000/xTUlMm4Y5xvvQr3TneDKEw
edg_wll_JobStatus: Connection refused: edg_wll_gss_connect(): server closed
the connection, probably due to overload
```

Les résultats des 10 simulations exécutées sur une période d'un mois environ sont donnés par la Figure 51.

Figure 51: Résultats des simulations de BLASTOG pour 10 jobs longs.

Nous remarquons d'après ces simulations, que BLASTOG dépend fortement de l'état de la grille de calcul. Et même avec un algorithme de gestion de la supervision des jobs, la grille continue à poser des problèmes de fiabilité qui se répercutent directement sur notre logiciel. Avec 5 simulations sur 10 terminées avec succès, BLASTOG reste fiable lorsque la grille est disponible. D'après les statistiques obtenues, une solution à ce problème consiste à lancer deux fois le même job sur la grille.

Enfin, pour comparer les résultats sur grille de calcul et les résultats sur Cluster pour notre échelle de données qui reste une échelle non adaptée à la grille de calcul, nous avons lancé les calcul pour les mêmes données sur le Cluster de l'ISIMA en utilisant un nombre de CPUs croissant. Les résultats sont donnés par la Figure 52. Nous remarquons qu'un speed-up de 51 est obtenu lorsque MPI-BLAST (Darling et al., 2003) est utilisé sur 128 CPUs. Nous remarquons

donc pour notre cas précis qu'un Cluster local de petite taille s'avère beaucoup plus performant que la grille.

Figure 52: Performances de MPI-BLAST sur la ferme de calcul du LIMOS.

Enfin, BLASTOG intègre la plupart des options de base de BLAST. En effet, l'utilisateur peut choisir la e-value, la taille du mot, le nom du programme, le nombre d'alignement affichés, le nom du fichier du sortie…etc. Il s'agit d'un logiciel totalement indépendant qui peut venir se greffer facilement à n'importe quel autre logiciel nécessitant une comparaison de séquences à grande échelle. Ses performances sont prometteuses. Il est adapté à des recherches massives de similarités pour des oligonucléotides destinés à être utilisés pou les puces à ADN. En revanche, pour des recherches nécessitant un temps de calcul de l'ordre de quelques jours (20000 séquences à 100000 séquences), une architecture de type Cluster est mieux adaptée. Des données plus importantes (quelques dizaines de millions à quelques milliards de séquences) nécessitant plusieurs années de calculs sont plus destinées à être traités pour la grille de calcul en espérant une meilleure fiabilité que celle constatée.

5. Conclusion

Dans le cadre de la conception de sondes pour biopuces à ADN destinées à une meilleure compréhension de la diversité microbienne et son impact sur l'environnement, l'un des défis majeurs est de traiter une masse très importante de données. L'ingénierie logicielle d'une part et le calcul haute-performance d'autre part représentent des clés essentielles pour l'exploitation à haut-débit de ces mines d'informations dans des temps raisonnables. L'ingénierie dirigée par les modèles permet à l'aide d'une démarche logicielle qui repose sur la méta-modélisation de représenter tous les aspects du développement d'un logiciel, de sa conception à son exploitation et ainsi proposer des solutions plus performantes et/ou plus précises grâce notamment à la génération de code. Les architectures parallèles telles que les fermes et les grilles de calcul apportent quant à elles une plus-value précieuse au niveau du gain du temps. Dans la suite, nous étudierons les applications développées durant cette thèse et qui exploitent ces technologies.

Chapitre IV : Applications et résultats

1. Introduction

Dans les parties précédentes, nous avons vu que la conception de sondes pour biopuce à ADN est une tâche à la fois complexe et consommatrice de ressources informatiques. Elle est complexe car elle intègre une multitude de paramètres, essentiels pour garantir la spécificité et la sensibilité des sondes avec également comme objectif la quantification. Quant au besoin de puissance de calcul, il est lié au caractère répétitif de certains programmes utilisés comme BLAST et ClustalW où la quantité de données à traiter est très importante, notamment quand il s'agit de traiter des traductions inverses complètes d'oligopeptides. L'ingénierie dirigée par les modèles est une méthode efficace pour simplifier au mieux cette complexité en garantissant une démarche standardisée. Elle ajoute en plus une couche d'abstraction précieuse : c'est le cas de la méta-programmation qui offre un niveau d'abstraction supplémentaire au développeur pour manipuler du code personnalisé. De plus, nous avons vu que les plates-formes parallèles telles les grilles de calcul, les clusters et les multiprocesseurs offrent des outils de choix pour augmenter la performance de ces algorithmes de conception d'oligonucléotides.

Dans ce chapitre seront présentés les développements réalisés concernant les algorithmes de conceptions de sondes et l'utilisation des différentes architectures distribuées, notamment la grille de calcul. De plus, seront présentées les différents développements effectués au laboratoire pour le traitement des données et la gestion des données biologiques en introduisant les nouvelles bases de données génomiques de références utilisées pour les logiciels de conception de sondes. Cependant, la difficulté de proposer un algorithme universel de conception de sondes ADN pour biopuces à ADN est due essentiellement à la complexité des données utilisées et au choix des critères de conception en fonction du type de puce à ADN désirée (Lemoine et al., 2009). La quantité des données génomiques est également un point crucial à prendre en compte dans cette démarche car elle ajoute un degré supplémentaire de complexité du point de vue performance de calcul. Nous proposons ici des solutions pour limiter la complexité et augmenter la performance de ces algorithmes en utilisant des solutions adaptées et en déployant les calculs coûteux sur les architectures distribuées à haute performance.

2. Algorithme de conception de sondes pour biopuces phylogénétiques sur la grille EGEE

Nous présentons ici les nouvelles approches développées concernant la sélection de sonde à ADN pour biopuces phylogénétiques et notamment l'utilisation de la grille de calcul pour la distribution du calcul à grande échelle. En effet, nous nous sommes basés sur nos développements antérieurs (Militon et al., 2007) pour proposer des solutions permettant une meilleure sélection des sondes en terme de sensibilité et de spécificité tout en augmentant la performance du logiciel.

A propos de la sensibilité, nous avons montré que les sondes PhylArray apparaissaient plus sensibles que celle sélectionnées avec PRIMROSE et dans la banque ARB. Elles pourraient donc potentiellement permettre d'accéder plus facilement aux phylotypes faiblement abondants présents dans les environnements étudiés. A propos de la spécificité, les sondes courtes PhylArray et ARB apparaissent les plus efficaces en engendrant moins d'hybridations croisées que les sondes longues PRIMROSE et PhylArray. De plus, les sondes sélectionnées avec le logiciel PhylArray permettent une vision exploratoire des communautés bactériennes étudiées contrairement aux autres outils de sélection qui ne génèrent que des sondes ciblant les fractions microbiennes connues. Pour cette raison, nous avons apporté à ce logiciel des améliorations au niveau de sa conception mais aussi au niveau performance afin d'obtenir des données encore plus fiables en un temps raisonnable.

2.1. Construction d'une base de données du biomarqueur phylogénétique: gènes codant la petite sous-unité d'ARNr.

Pour tout logiciel de conception de sonde ADN il existe une base de données de référence. Ici, les données utilisées pour cette base sont des données extraites à partir de la banque de données internationale EMBL. Il s'agit des données concernant en particulier les microorganismes appartenant aux procaryotes (PRO) et aux champignons (FUN). L'étude de ces microorganismes par des biopuces phylogénétiques permet de comprendre leur évolution et d'étudier leur développement dans les environnements cibles. L'idée ici est de réaliser la conception de sondes représentatives des genres ou des espèces (et des niveaux taxonomiques supérieurs) de ces microorganismes. Nous obtiendrons ainsi une base de données de sondes pour

biopuces à ADN de type « PhyloChip ». Une première génération de biopuces de près de 4000 sondes avait déjà était conçue en ciblant respectivement 1373 genres de *Bacteria* et 86 d'*Archaea*.

2.1.1. Construction de la base de données

Pour obtenir une base de données représentative des communautés microbiennes procaryotes et champignons, un logiciel d'extraction des données à partir de la banque EMBL a été développé. L'objectif du prétraitement des données est de construire une base de données plus exhaustive que celles proposées dans la littérature [ARB (Ludwig et al., 2004), GreenGenes[1] et Ribosomal Database Project[2]]. Plusieurs étapes ont été nécessaires pour l'obtention des données. Ces données sont classées par genre, chaque genre correspond à un fichier de séquences nucléiques au format FASTA. C'est le champ 'OC' des fiches EMBL correspondants aux mots clés « rel_std_pro » et « rel_std_fun » (cf. partie III.2.1) qui détermine la classification taxonomique et donc le genre de chaque séquence extraite en se basant sur la base de classification phylogénétique de GenBank. L'espèce désignée par le champ 'OS' est reportée sur la ligne de commentaire dans le fichier FASTA correspondant au genre. Une séquence extraite de la base de données est sauvegardée si elle vérifie tous les critères suivants :

- Le pourcentage des bases inconnues (N) est inférieur à 10%.
- Elle ne contient pas de « stretch [3] » de 10 bases inconnues (N).
- Sa longueur est comprise entre 1200 pb et 1600 pb.
- Elle contient un des motifs « 16S, 16s, rRNA, Small subunit » pour les procaryotes.
- Elle contient un des motifs « 18S, 18s, rRNA, Small subunit » pour les champignons ;

Et elle est rejetée sinon

Ensuite, la base construite subit une succession de traitements pour le filtrage et la sélection des données de bonne qualité. En effet, les données existantes dans les banques de données internationales, telle que EMBL, présentent souvent une disparité, des erreurs et des redondances (Chen, 2005).

2.1.1.1. Suppression des séquences redondantes

A la fin de la création des fichiers des genres, chaque fichier est traité pour supprimer les séquences redondantes. Une séquence est redondante si il existe dans le même fichier une autre

[1]GreenGenes: http://greengenes.lbl.gov/cgi-bin/nph-index.cgi
[2]RDP: Ribosomal Database Project: http://rdp.cme.msu.edu/
[3]Stretch: Sous-séqeunce comportant une suite du même acide nucléique

séquence identique et/ou ayant le même numéro d'accession. Dans ce cas, la séquence est supprimée ou autrement dit n'est pas intégrée dans le fichier. Par conséquent, nous obtenons un fichier contenant des séquences rRNA différentes les unes des autres tout en gardant l'information d'origine des séquences identiques ce qui évite d'alourdir le traitement par le simple fait de la redondance de séquences .

2.1.1.2. Elimination des séquences chimériques et mal annotées

Cette étape permet d'identifier les séquences artéfactuelles (chimériques) et les séquences qui ont été assignées par erreur à un groupe taxonomique. Cependant, il est difficile de dire de façon définitive qu'une séquence appartient ou pas à un groupe donné. Nous avons choisi d'utiliser la méthode de classification K-MEANS[1] permettant de partitionner un groupe d'observations statistiques (ici des vecteurs) suivant leurs degrés de similarités d'un point de vue d'une distance prédéfinie. Le principe est le suivant: soit $O = \{x_1, \ldots, x_n\}$ l'ensemble des observations, où chaque observation est un vecteur appartenant à l'espace vectoriel à d dimensions $\mathfrak{R}^l$, la méthode a pour objectif de partitionner les n observations en k clusters ($k < n$) $S = \{s_1, \ldots, s_k\}$ de manière à minimiser la sommes des carrées des groupes :

$$argmin_s \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

Où μ_i est la moyenne de S_i.

Dans notre cas, l'étape commence par la construction d'une matrice de vecteurs. Chaque vecteur est la représentation de la similarité relative d'une séquence par rapport à toutes les autres séquences du groupe en utilisant le programme BLAST. Ces vecteurs sont ensuite considérés comme l'ensemble des observations O de la méthode K-MEANS. La dimension de l'espace vectoriel de travail est égale au nombre de séquences du groupe courant (fichier). Ici, nous distinguons deux cas uniques possibles : la séquence appartient au groupe homogène en question ou n'y appartient pas. Le nombre de cluster est égal à 2. Cette méthode donne des résultats très satisfaisants puisqu'elle permet d'écarter les séquences qui n'appartiennent visiblement pas au groupe taxonomique. En effet, seules des séquences fortement homogènes sont gardées dans le fichier correspondant au genre. Par exemple, pour le genre *Ambrosiozyma*, parmi les 6 séquences formant le groupe et ayant les numéros d'accession EU011670, EU011671, EU011673, EU011674, EU011675 et EU011676, les deux séquences EU011670 et EU011671 ont été écartées car elles présentaient au sens K-MEANS des différences avec le reste du groupe. Si aucune séquence n'est identifiée comme séquence à éliminer alors le groupe reste inchangé.

[1]K-MEANS : méthode de classification non supervisée à base de nuée dynamique

2.1.1.3. Réorientation des séquences

L'une des étapes théoriquement impensable de la création de notre base phylogénétique pour puces à ADN est la vérification de l'orientation des séquences dans les fichiers genres. En effet, en utilisant le programme OrientationCheker (Ashelford et al., 2006), certains fichiers présentaient des séquences orientées dans le sens contraire de la majorité des séquences du groupe. Cela est dû essentiellement à une mauvaise maîtrise des connaissances de biologie moléculaire et donc à des erreurs grossières commises par les personnes ayant générées ces données. Par exemple, la séquence ayant le numéro d'accession DQ345280, est affiliée parmi les *Saccharomyces* dans la banque EMBL comme étant « *Saccharomyces sp. WW-W23 18S ribosomal RNA gene, partial sequence.*» ; mais en effectuant un alignement BLAST avec l'espèce *Saccharomyces naganishii* ayant le numéro d'accession AB016512 (Figure 53), on réalise qu'elle présente un score très important mais que l'alignement s'est fait sur le brin négatif comme l'indique la figure IV.1. Une telle séquence mal orientée peut causer des problèmes au niveau de l'alignement global final car c'est la séquence inverse complémentaire de cette séquence DQ345280 qui a du être répertoriée dans la banque EMBL.

Figure 53: Similarités entre les séquences DQ345280 et AB016512.

Nous avons alors développé un programme « ToSens.pl » qui permet de remettre les séquences dans la bonne orientation et de les conserver dans la base de données afin d'obtenir des fichiers homogènes. Cette étape de réorientation permet de comparer chaque séquence par rapport à une séquence de référence (*Bacteroides intestinalis* AB437413 pour les procaryotes et EF014366 *Blastocladiella emersonii* pour les champignons) afin d'effectuer la réorientation. L'étape suivante permet de créer des sous-groupes taxonomiques très homogènes permettant un alignement optimisé des séquences.

2.1.1.4. Construction de sous-groupes homogènes de séquences

Afin de créer des sous-groupes de séquences homogènes pour faciliter les alignements, nous avons développé un programme « *BlastClustPhyl.pl* » permettant de fragmenter un fichier de séquences d'un même genre. Pour ce faire, nous avons utilisé le programme BlastClust[1] (basé sur MegaBlast[2] pour la recherche de similarité) pour d'abord créer les sous-groupes (cluster) représentés par les séquences les plus proches. Les paramètres utilisés étaient:

-b F (prendre en compte un seul membre d'une paire de séquence pour le threshold)
-S 97 (pourcentage d'identité entre les séquences)
-L .98 (pourcentage de la longueur utilisée)
-p F (séquences nucléiques)

Une étape finale consiste à éliminer les sous-groupes trop éloignés de la majorité des séquences constituant le genre. Les sous-groupes représentant moins de 0,8% des séquences ou n'étant représentés que par une séquence ne sont pas conservés ce qui est rarement le cas. Pour les groupes comptant un très grand nombre de séquences (quelques centaines à quelques milliers), la construction des sous-groupes permet de réduire la complexité d'un genre pour faciliter l'alignement multiple dans les étapes ultérieures.

2.1.1.5. Alignements multiples des séquences du même genre

Il s'agit de la dernière étape permettant de déterminer la séquence consensus d'un genre à la base de la détermination des sondes. Cette étape est très coûteuse en termes de temps calcul. En effet, après création des sous-groupes, nous avons obtenu des fichiers de séquences intrinsèques représentatives de leurs genres respectifs. Afin d'exploiter notre base de données, il faut que celle-ci soit présentée au logiciel de sélection de sondes sous forme de séquences consensus (Militon et al., 2007) pour identifier les oligonucléotides qui seront déposés sur la biopuce. Un alignement multiple permet donc d'aligner ensemble les séquences homogènes des sous-groupes. Nous obtenons pour chaque genre un ou plusieurs fichiers d'alignement. Par souci de performance et de qualité des alignements effectués, nous avons évalué plusieurs logiciels d'alignement multiple de séquences. Quatre logiciels n'ont pas pu être installés : MAVID (Bray et Pachter, 2004), KAlign-KalignVu, MUMSA (Lassmann et Sonnhammer, 2006) et FSA (Bradley, 2009). Le tableau IV.1 récapitule l'ensemble des tests réalisés sur une machine Dual-Core AMD Opteron 1.8GHz, 128Go de RAM. Les calculs ont été faits de manière séquentielle sur un fichier contenant 3716 séquences du genre *Bacillus*. Les logiciels qui ont été comparés sont présenté dans

la Figure 54: PCMA (Pei et al., 2003), ClustalW (Thompson et al., 1994), POA (Lee et al., 2002), OPAL (Wheeler et Kececioglu, 2007), T-COFFEE (Notredame et al., 2000). L'évaluation de la qualité de l'alignement a permis de sélectionner le programme ClustalW donnant le meilleur compromis en termes de qualité et de rapidité d'alignement. En effet, la qualité des alignements donnés par le programme est la plus satisfaisante même si sa performance se place dans la moyenne des logiciels testés.

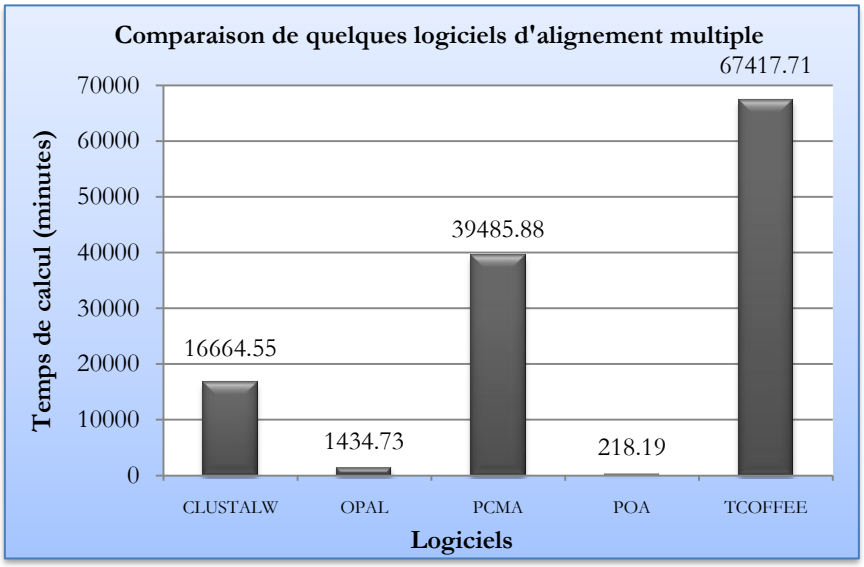


Figure 54: Alignement multiple de 3716 séquences avec les différents logiciels testés.

Nous avons réalisé des tests de performance de ClustalW ainsi que de sa version parallèle ClustalW-MPI (Li, 2003). Les autres logiciels n'ayant pas été parallélisés. Nous avons tout d'abord effectué les tests d'alignement sur des fichiers dont le nombre de séquences variait entre 24 et 86 pour avoir le comportement sur des données de quelques dizaines de séquences. Nous avons comparé les performances de ClustalW sur une machine SMP[1] avec 8 Quad-Core AMD Opteron à 2.3GHz avec 256 Go RAM (mais en utilisant dans un premier temps un seul cœur) avec la version disponible sur le site de l'EMBL-EBI. Les résultats sont présentés dans la Figure 55. Nous remarquons d'après les performances, que la version en ligne est plus rapide que celle installée en local sur nos machines. Les calculs étant lancés avec la même version du programme et avec les mêmes options par défaut. Cette version en ligne s'exécute sur des machines bien plus performantes à l'EBI.

[1]SMP: Symmetric Multiprocessor

Figure 55: Performances du ClustalW version 2.0.11 en fonction du nombre de séquences

Nous avons ensuite évalué les performances pour les mêmes fichiers en utilisant la version MPI sur le la grille locale du LIMOS composée d'une ferme de calcul et de 3 machines SMP. La ferme de calcul est composée d'un nœud Maitre et de 22 nœuds esclaves. Les nœuds sont des machines avec chacune 8 cœurs (Bi-Quad-Core) et 10 Go de RAM. Les SMPs sont des machines avec 32 cœurs (Octo-Quad-Core) et 256Go de RAM. La capacité totale est de 272 coeurs. L'architecture globale est présentée dans la Figure 56.

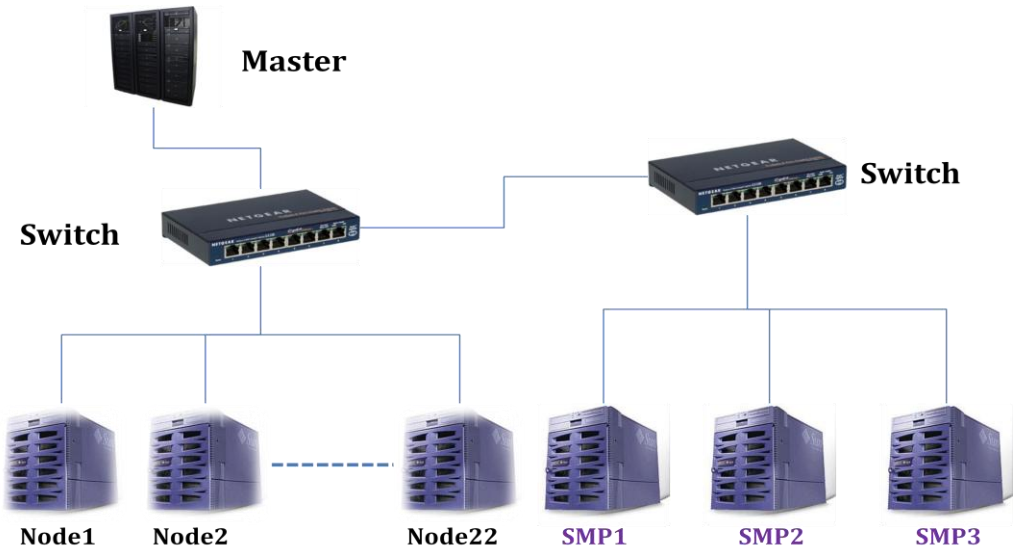


Figure 56: Architecture de la grille locale du LIMOS.

Les résultats sont montrés sur la troisième courbe de la Figure 55 en utilisant 20 cœurs. Les calculs ont été réalisés alors sur les 3 premiers nœuds. Les Speed-up de la version parallèle de ClustalW par rapport à la version locale sont présentés dans le Tableau 14.

Tableau 14: Speed-up de ClustalW-MPI pour différents fichiers sur 20 cœurs de la ferme de calcul.

Genre	Nombre de séquences	Speed-up
Scutellospora	**24**	3.1
Teratosphaeria	**30**	2.1
Penicillium	**45**	3.4
Glomus	**55**	5.0
Aspergillus	**86**	5.7

Bien évidemment, le nombre de séquences influence directement la performance des différents programmes. De plus, le speed-up de la version MPI par rapport à la version locale augmente lorsque le nombre de séquences du fichier augmente. Plus les données sont importantes, plus l'utilisation de ClustalW-MPI est nécessaire pour garantir une meilleure performance. Nous avons testé le programme pour 86 séquences sur la ferme de calcul en faisant varier le nombre de cœurs utilisés. Les résultats sont présentés dans la Figure 57. Nous observons un speed-up relativement faible comparé au nombre de cœurs. C'est principalement dû à l'utilisation d'une parallélisation avec MPI et donc aux communications (loi d'Amdahl). Nous remarquons cependant une diminution intéressante dans le temps de calcul lorsque le nombre de cœurs utilisés augmente.

Figure 57: Performance de ClustalW-MPI sur la ferme de calcul du LIMOS.

Ensuite, nous avons choisi d'utiliser la ferme de calcul pour réaliser les alignements à l'aide de ClustalW-MPI. La Figure 58 montre les performances de ClustalW-MPI sur la ferme de calcul en utilisant 100 cœurs en augmentant progressivement le nombre de séquences du fichier d'entrée. Nous constatons qu'avec un nombre de séquences équivalent (quelques milliers de séquences), le temps d'exécution devient bien plus intéressant qu'avec des fichiers de plus petites tailles (quelques dizaines de séquences). C'est-à-dire que le speed-up est plus important lorsqu'on travaille sur des données de grande taille ce qui est bien adapté à notre problème. Par exemple, nous avons besoin de 6 jours et demi environ pour obtenir le résultat d'alignement de 3000 séquences sur une machine de 1.83 Ghz 128 Go de RAM. Alors qu'en exécutant ce même fichier sur une ferme de calcul en utilisant 100 cœurs (voir la figure 58) on obtient le même résultat en 3 heures et 12 minutes environ. Un speed-up de 60 est alors obtenu.

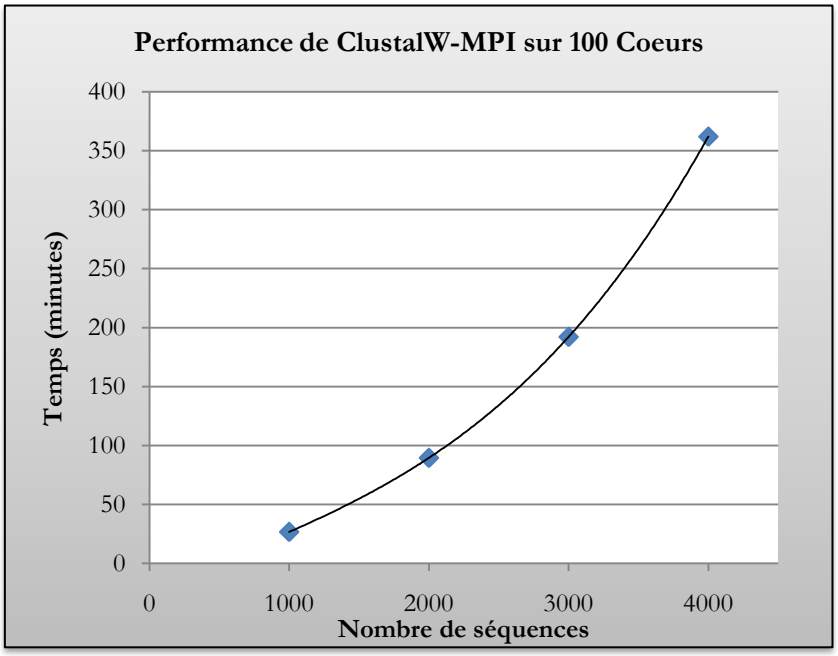


Figure 58: Comportement du ClustalW-MPI sur la ferme de calcul.

De part le nombre de séquences à aligner nous avons opté pour une stratégie d'alignement d'alignements permettant réduire la complexité d'alignement et de de diminuer les temps de calcul. Notre base contient au total de 4965 séquences de champignons et 39 807 séquences de procaryotes qui correspondent respectivement à 1588 fichiers et 2696 fichiers.

Le programme OPAL a été sélectionné car il permet également de réaliser un alignement d'alignements en un temps très court (4 à 12 secondes, quelque soit le nombre de séquences). A titre d'exemple, nous avons choisit le genre *Aspergillus* pour démontrer l'intérêt d'utiliser OPAL afin de réaliser l'alignement des séquences. Les résultats sont donnés dans la Figure 59. Nous

remarquons en effet que lorsque l'alignement de toutes les séquences avec ClustalW nous obtenons un temps de calcul supérieur par rapport à l'approche du programme OPAL. Nous avons alors développé un outil qui utilise OPAL et qui traite récursivement l'ensemble des fichiers d'alignement générés. Le résultat pour le genre *Aspergillus* est de 420 secondes si on aligne séparément les 2 sous-groupes puis en les regroupant (2 secondes ont été nécessaire pour le regroupement), contre 779 secondes en alignant en une seule fois le fichier entier de 86 séquences. Le gain est alors de 50% du temps de calcul. Le processus a alors été appliqué à tous les genres.

Figure 59: Comparaison entre l'alignement multiple de 86 séquences d'*Aspergillus* et l'alignement d'alignement avec Opal de ses deux sous-groupes formés de 52 et de 34 séquences.

2.1.1.1. Fusion des sous-groupes

Les sous-groupes de chaque genre ont été regroupés en un seul groupe en utilisant OPAL. Puis les fichiers ont été retransformés au format CLUSTAL pour l'utilisation dans l'algorithme PhylArray de conception de sondes. Après regroupement, nous avons obtenu 1441 fichiers champignon et 1907 fichiers procaryotes. Il s'agit de la dernière partie de la création de la base de données phylogénétique pour la conception de sondes. Le temps total de fusion était d'environ 1 heure.

2.1.2. Gestion des données

Il existe deux composantes fondamentales pour le logiciel présenté ici : la première concerne la base de données de référence qui regroupe toutes les séquences au format FASTA des espèces appartenant à un même genre et la deuxième est composée des fichiers d'alignements par genre pour les procaryotes et les champignons. L'utilité de la deuxième partie réside dans la disponibilité immédiate du fichier d'alignement pour effectuer la conception des sondes. En effet, il n'est plus utile de réaligner les séquences à partir de la base de données de référence pour chaque requête. La base de données est donc composée d'un fichier FASTA de 7 Mo pour les champignons, d'un fichier FASTA de 110 Mo pour les procaryotes, de 1441 fichiers d'alignements pour les champignons et de 1907 fichiers d'alignements pour les procaryotes.

2.2. Algorithme de conception de sondes

Le logiciel présenté ici est un logiciel de conception de sondes oligonucléotidiques pour biopuces phylogénétique (PhylArray). Il permet de déterminer pour un genre donné, les oligonucléotides qui vérifient les critères suivants implémentés dans le logiciel :

- **La longueur de sonde** : c'est la longueur des oligonucléotides qui seront sur la puce.
- **Le seuil de spécificité** (threshold): c'est le seuil de similarité au dessus duquel une séquence est considérée comme provoquant une hybridation croisée.
- **Le nombre maximum d'hybridations croisées** : est le nombre autorisé d'hybridations croisées.

La première étape est basée sur la construction de séquence consensus à partir de l'alignement CLUSTAL des séquences d'un genre préalablement obtenu. La séquence consensus utilise le code IUB[1]. Par exemple, si l'on considère le genre *Yarrowia*, nous avons obtenu l'alignement suivant (une fenêtre pour l'exemple):

```
AB018158|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
DQ486711|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
DQ438177|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
EU434621|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
DQ438182|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
EF190312|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
DQ437080|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTTAT
DQ437079|       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
                ********************************
```

La séquence consensus est donnée par :

```
CONSENSUS       TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGWKRK
```

Les autres séquences (seqx) sont utiles pour garantir un aspect exploratoire de nos biopuce :

```
Seq1        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAGGG
Seq2        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTTAT
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAAAA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAAAC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAAGA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGAAGC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGACAA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGACAC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGACGA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGACGC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTAAA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTAAC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTAGA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTAGC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTCAA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTCAC
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTCGA
Seqx        TGAATGGTTTAGTGAGACCTTGGGAGGGCGAGATGTCGC
```

Cette sélection permet de garantir de cibler le groupe (le genre) ou un niveau taxonomique plus haut (famille, ordre, etc.…) pour lequel on réalise la conception de sondes. D'un point de vue expérimental, le spot contant les oligonucléotides dégénérés sera composé d'un mélange de tous les oligonucléotides possibles. L'utilisateur obtient alors une sortie contenant la séquence dégénérée et les séquences spécifiques correspondantes utilisées pour la conception, ainsi il peut construire des biopuces avec des oligonucléotides dégénérés et/ou des oligonucléotides spécifiques.

2.2.1. Les étapes de la conception

L'algorithme proprement dit est composé, entre autre, de 2 principales étapes qui seront détaillées dans la suite :

La fragmentation de la séquence consensus

La recherche de spécificité pour les hybridations croisées

2.2.1.1. Fragmentation de la séquence consensus

Il s'agit ici de générer à partir de la séquence consensus construite auparavant tous les oligonucléotides dégénérés. Le nouvel algorithme décompose la séquence en plusieurs fragments (fenêtres en bleu dans le schéma de la Figure 60) pour la parallélisation du calcul. On obtient alors pour chaque fragment correspondant aux positions de début des oligonucléotides une tâche

[1] IUB: International Union of Biochemistry

qui sera exécutée sur la grille. Cela permet alors de paralléliser le calcul. Par exemple, pour créer 2 tâches de la séquence consensus de 69 mers suivante, on obtient la décomposition de la Figure 60.

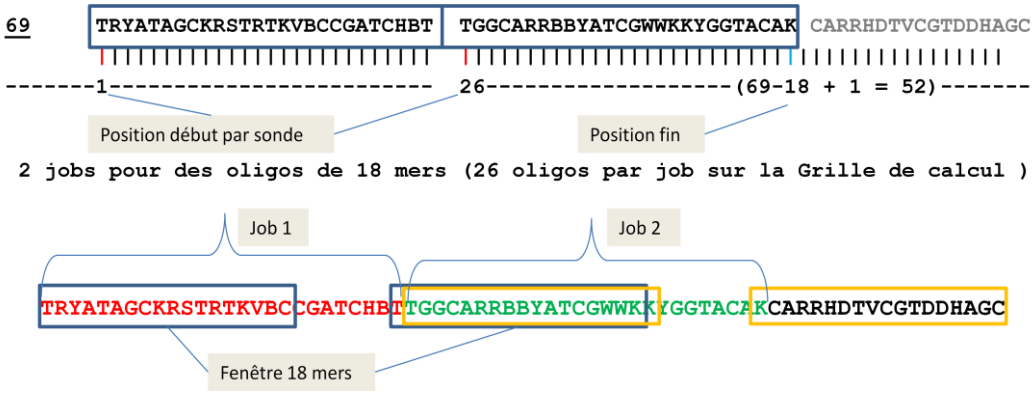


Figure 60: Exemple de décomposition d'une séquence en sondes 18 mers.

Il s'agit donc de reconstruire la conception complète en recomposant les différentes sous-séquences. En effet, chaque tâche traite un fragment de la séquence en recherchant les hybridations croisées pour chaque sonde potentielle contre la base de référence exceptée le genre en cours de conception.

2.2.1.2. La recherche de spécificité

Cette partie permet de tester pour toutes les sondes générées pour une position donnée les hybridations croisées potentielles en utilisant les paramètres choisis par l'utilisateur (en particulier, le seuil de spécificité et le nombre d'hybridations croisées). Le test de spécificité se fait en comparant à la base de données les oligonucléotides spécifiques avec le programme BLAST. Les paramètres utilisés sont la taille du mot (W=7), le filtrage de la complexité (F=F) et l'e-value égale à 100. Les critères de Kane (75% d'identité avec les cibles potentielles et une séquence continue de 15 nucléotides représentent une hybridation croisée) sont ensuite appliqués aux sondes. Seules les sondes vérifiant ces critères sont conservées.

La Figure 61 récapitule les différentes étapes de la conception de sondes à l'aide de notre logiciel. La comparaison BLAST n'est réalisée que contre le reste de la base de données pour un genre donné.

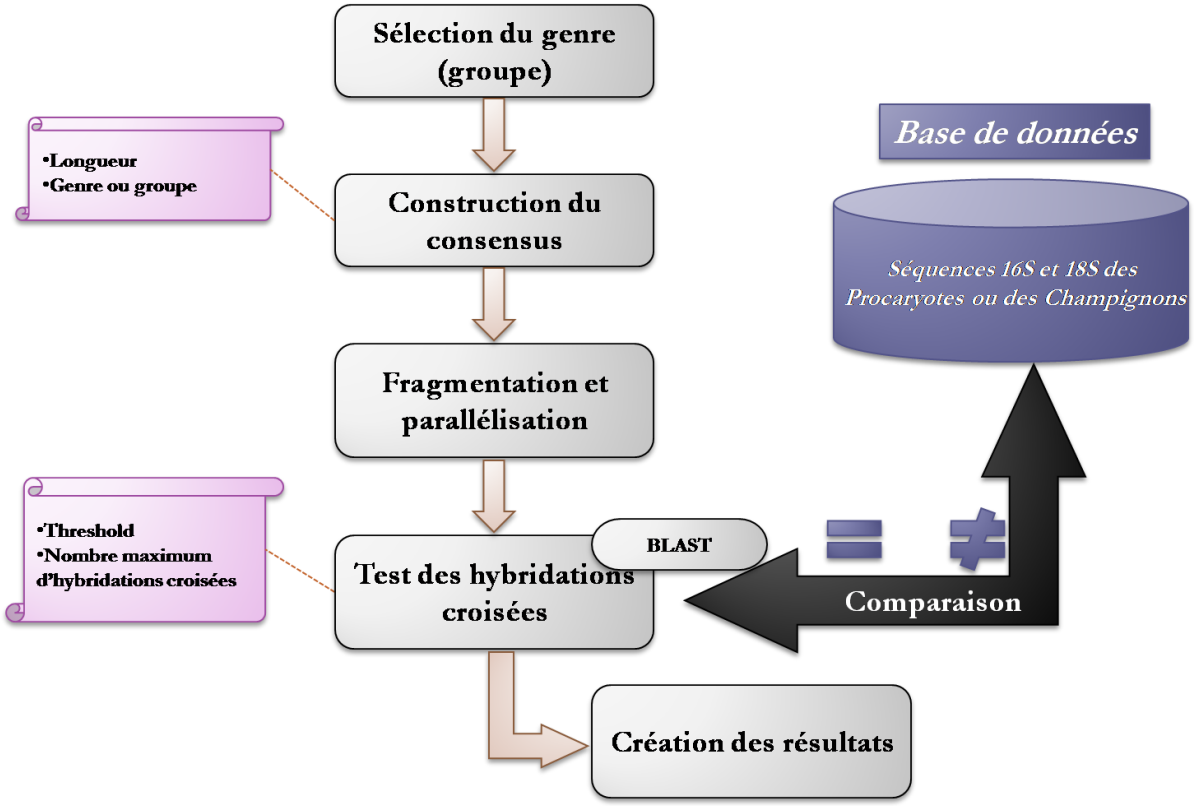


Figure 61: Etapes de conception de sondes avec le logiciel PhylArray.

2.2.2. Matériels et architecture

Le logiciel implémentant notre nouvel algorithme est une application Web développée avec PHP 5 et utilise des scripts Perl pour la conception de sondes. Elle utilise une base de données de conception et de gestion d'utilisateur sous MySQL 5.1. L'application est installée sur une UI de la grille de calcul EGEE entièrement configurée au laboratoire. Il s'agit d'une machine virtuelle installée sous Xen (www.xen.org) reposant sur 2 cœurs à 1.8 GHz et avec 1 Go de RAM. L'utilisation du logiciel se fait avec un accès par « login » et mot de passe. Les calculs sont entièrement faits sur la grille EGEE notamment la recherche de spécificité.

L'interface permet de sélectionner les genres en choisissant de faire une seule conception ou de soumettre une liste de conceptions à la grille de calcul comme le montre la Figure 62. L'utilisateur est capable de choisir également la base de données de référence sur laquelle s'effectuera la recherche de spécificité.

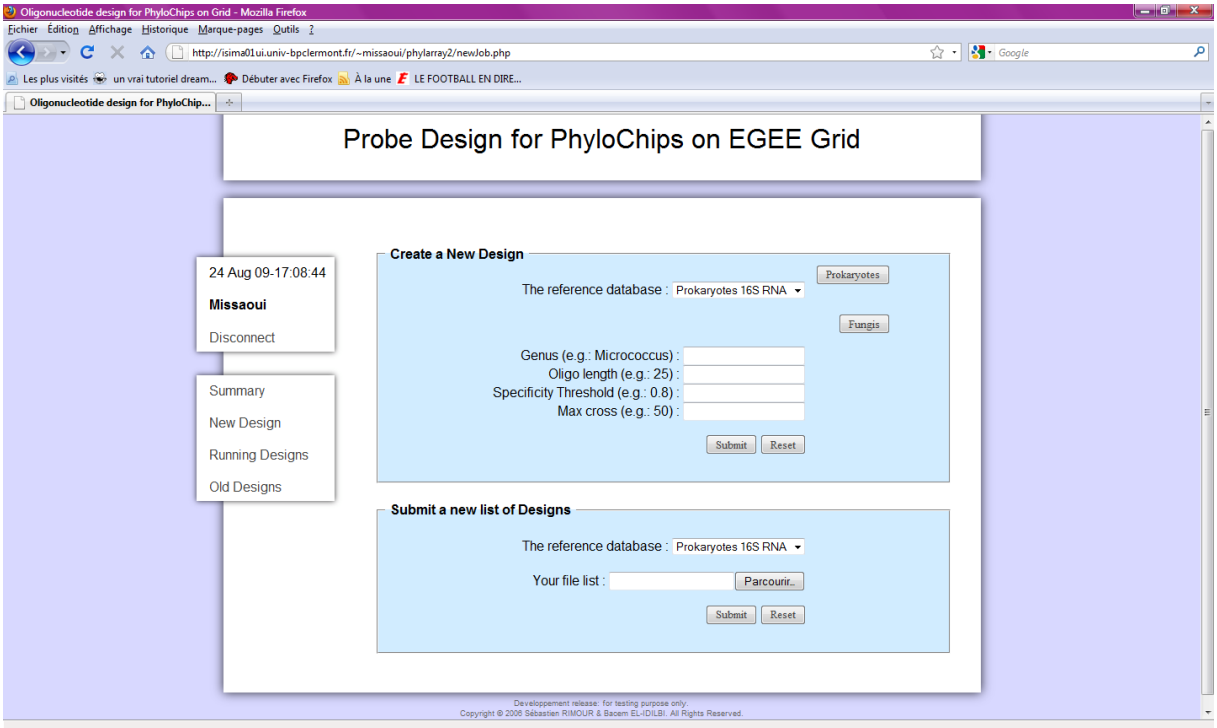


Figure 62: Interface de sélection des critères de conception de sondes pour biopuces ADN phylogénétiques sur la grille EGEE.

Les différents stades de conceptions des sondes sur la grille sont affichés à l'utilisateur suivant leurs états respectifs (en cours d'exécution ou terminée). Le résultat est disponible dès que la conception est terminée et peut être téléchargée à partir de l'interface. Afin d'optimiser le temps de calcul, nous avons parallélisé l'application sur la grille EGEE et nous l'avons testée pour tous les genres des champignons.

2.3. Distribution du calcul sur la grille EGEE

Afin d'optimiser les temps de calcul pour la conception de sondes ADN pour les biopuces phylogénétiques, nous avons développé un module de parallélisation des calculs sur la grille EGEE. La parallélisation se fait lors de la fragmentation de la séquence consensus avant la recherche de spécificité. Cette dernière est en effet réalisée sur la grille. Afin de profiter de la taille de la grille européenne, plusieurs conceptions peuvent être lancées de manière simultanée.

2.3.1. Authentification sur la grille

La parallélisation sur grille de calcul nécessite des autorisations d'accès aux ressources informatiques de la grille EGEE. Le CA (Certificate Authority) fournit à chaque utilisateur de la grille EGEE inscrit à une VO (Virtual Organization) un certificat composé d'une clé publique et d'une clé privée. Les deux clés sont générées à partir d'une seule clé au format PKCS12 (Public Key Cryptographic Standards) (ayant une extension .p12) – qui définit un format de fichier utilisé

pour stocker la clé privée et le certificat de clé publique correspondant en les protégeant par un mot de passe. Par exemple la commande suivante permet de générer les deux clés pour une utilisation de la grille:

```
[missaoui@isima01UI ~]openssl pkcs12 -in mohieddine.p12 -clcerts -nokeys -
out $HOME/.globus/usercert.pem
[missaoui@isima01UI ~]
[missaoui@isima01UI ~]openssl pkcs12 -in mohieddine.p12 -nocerts -out
$HOME/.globus/userkey.pem
```

Les deux fichiers générés (.pem) sont utilisés ensuite pour générer le proxy nécessaire à l'authentification de l'utilisateur sur toutes les machines de la grille (WMS, CE, SE, RB, BDII…) à l'aide de la commande suivante:

```
[missaoui@isima01UI ~]voms-proxy-init --voms biomed
Enter GRID pass phrase:
Your identity: /O=GRID-FR/C=FR/O=CNRS/OU=LIMOS/CN=Mohieddine Missaoui
Creating temporary proxy
................................................... Done
Contacting  cclcgvomsli01.in2p3.fr:15000 [/O=GRID-FR/C=FR/O=CNRS/OU=CC-
LYON/CN=cclcgvomsli01.in2p3.fr] "biomed" Done

Creating proxy ..................................... Done

Warning: your certificate and proxy will expire Mon Feb  8 15:45:00 2010
which is within the requested lifetime of the proxy
[missaoui@isima01UI ~]
```

Nous avons utilisé pour les développements réalisés les ressources de la VO biomed. La commande suivante permet de lister toutes les ressources disponibles ainsi que leur charge de calcul. Les ressources disponibles sont les CE, les SE ou les RB.

```
[missaoui@isima01UI ~]lcg-infosites –vo biomed ce
#CPU    Free    Total Jobs      Running Waiting ComputingElement
----------------------------------------------------------
 124      60       0              0        0
lcg38.sinp.msu.ru:2119/jobmanager-lcgpbs-biomed
2718      16       0              0        0    ce04-
lcg.cr.cnaf.infn.it:2119/jobmanager-lcglsf-debug
   0       0       0              0     444444  ppsce03.pic.es:8443/cream-
condor-condor
 421     113       0              0        0
polgrid1.in2p3.fr:2119/jobmanager-pbs-sdj
1346       1       0              0        0
gridce.pi.infn.it:2119/jobmanager-lcglsf-grid4
1136     874       3              2        1
trekker.nikhef.nl:2119/jobmanager-pbs-qshort
1776    1744       0              0     444444
svr021.gla.scotgrid.ac.uk:2119/jobmanager-lcgpbs-q3d
  20       4      12              6        6    ce001.grid.uni-
sofia.bg:2119/jobmanager-lcgpbs-biomed
/…/
[missaoui@isima01UI ~]lcg-infosites –vo biomed se
Avail Space(Kb) Used Space(Kb)  Type    SEs
----------------------------------------------------------
1954042367      210476364       n.a     fornax-se.itwm.fhg.de
319315555       82124819        n.a     se02.marie.hellasgrid.gr
```

```
30387761434     164970612257    n.a     grid-cert-03.roma1.infn.it
4377513251      52782116        n.a     se.pakgrid.org.pk
1110552301      147250237       n.a     se01.cat.cbpf.br
1099368480      540             n.a     glite03-kvm.hpc2n.umu.se
27813281178     142518029       n.a     gw-3.ccc.ucl.ac.uk
6572167         1951418068      n.a     se01.grid.auth.gr
/…/
[missaoui@isima01UI ~]lcg-infosites –vo biomed wms
https://lcgrb02.jinr.ru:7443/glite_wms_wmproxy_server
https://svr022.gla.scotgrid.ac.uk:7443/glite_wms_wmproxy_server
https://wms.pnpi.nw.ru:7443/glite_wms_wmproxy_server
https://grid-wms2.desy.de:7443/glite_wms_wmproxy_server
https://lcgrb01.jinr.ru:7443/glite_wms_wmproxy_server
https://wms01.grid.sinica.edu.tw:7443/glite_wms_wmproxy_server
https://grid-wms14.desy.de:7443/glite_wms_wmproxy_server
https://gridrb.fe.infn.it:7443/glite_wms_wmproxy_server
https://wmslb101.grid.ucy.ac.cy:7443/glite_wms_wmproxy_server
/…/
[missaoui@isima01UI ~]
```

Comme nous pouvons le remarquer, il y a des informations utiles sur la disponibilité des éléments de calcul et la capacité de stockage des SE ou encore les WMS disponibles. D'autre part, il est important de prendre en considération les logiciels qui sont utilisés dans la conception de sonde et en particulier le programme BLAST. En effet, nous avons choisi de copier les exécutables compilés sur les SE de la grille en faisant des réplicas (sachant que toutes les plateformes de la grille EGEE sont identiques). Nous avons alors utilisé les commandes LCG[1] suivantes:

```
[missaoui@isima01UI ~]lcg-cr –d grid-cert-03.roma1.infn.it –-vo biomed
lfn:/grid/biomed/missaoui/logiciels/blast.tar $HOME/blast.tar

[missaoui@isima01UI ~]
```

Enfin, pour le bon fonctionnement de l'application, il est important de vérifier la syntaxe des fichiers JDL ainsi que l'exécution des scripts utilisés pour la recherche de spécificité. Un job est soumis à l'aide de la commande suivante:

```
[missaoui@isima01UI ~]glite-wms-job-submit –a job_4536_0.jdl

Connecting to the service
https://grid25.lal.in2p3.fr:7443/glite_wms_wmproxy_server

====================== glite-wms-job-submit Success ======================

The job has been successfully submitted to the WMProxy
Your job identifier is:

https://grid09.lal.in2p3.fr:9000/s65Bt_cYDObWt-Grh2em2A
[missaoui@isima01UI ~]
```

Pour vérifier enfin le statut du job soumis et récupérer les résultats directement et pour simplifier, nous exécutons la commande suivante:

```
[missaoui@isima01UI ~]glite-wms-job-status
https://grid25.lal.in2p3.fr:7443/glite_wms_wmproxy_server
```

[1]LCG: LHC Computing Grid

```
**************************************************************
BOOKKEEPING INFORMATION:

Status info for the Job : https://grid09.lal.in2p3.fr:9000/s65Bt_cYDObWt-
Grh2em2A
Current Status:     Scheduled
Status Reason:      Job successfully submitted to Globus
Destination:        lcgce02.gridpp.rl.ac.uk:2119/jobmanager-lcgpbs-grid500M
Submitted:          Wed Aug 26 16:40:39 2009 CEST
**************************************************************

[missaoui@isima01UI ~] glite-wms-job-output
https://grid25.lal.in2p3.fr:7443/glite_wms_wmproxy_server

Connecting to the service
https://grid25.lal.in2p3.fr:7443/glite_wms_wmproxy_server

================================================================================

                          JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
https://grid09.lal.in2p3.fr:9000/s65Bt_cYDObWt-Grh2em2A
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/missaoui_s65Bt_cYDObWt-Grh2em2A

================================================================================
[missaoui@isima01UI ~]
```

Le déploiement repose en grande partie sur le développement d'API génériques qui décharge l'utilisateur de toutes ces commandes et qui permet de superviser automatiquement les jobs.

2.3.2. Déploiement et parallélisation

Pour chaque fragment de la séquence consensus, plusieurs fichiers de configuration propres à la grille de calcul sont créés : Les scripts d'exécution, les fichiers JDL (Job Description Language), et les fichiers des identifiants des jobs. Nous obtenons par exemple pour le premier fragment de la séquence consensus du genre *Teratosphaeria* :

Le fichier JDL:

```
PerusalFileEnable = true;
PerusalTimeInterval = 120;
Executable="/bin/bash";
RetryCount = 3;
Arguments = "j_4389_0.sh";
StdOutput = "j_4389_0.out";
StdError = "j_4389_0.err";
InputSandBox = {"puceenv.pl","Teratosphaeria.aln","j_4389_0.sh"};
OutputSandbox =
{"j_4389_0.out","j_4389_0.err","result_Teratosphaeria0.txt","Teratosphaeria
0.tmp"};
```

Le script de conception de sonde:

```
rm -rf blast/
mkdir blast
cd blast/
lcg-cp lfn:/grid/biomed/missaoui/tmp/blast.tar file:./blast.tar
tar -xvf blast.tar
cd ..
lcg-cp lfn:/grid/biomed/missaoui/tmp/4389_arn.tar file:./4389_arn.tar
rm -f arn*
ls -al arn*
tar -xvf 4389_arn.tar
perl puceenv.pl Teratosphaeria 0 333 25 0.8 50
rm -f 4389_arn.tar
rm -f arn.txt*
rm -rf blast/
```

Le fichier de l'identifiant de job:

```
###Submitted Job Ids###
https://lapp-lb01.in2p3.fr:9000/5CgYCLU2b8iKZ-raSBEtYg
```

La base de données de référence est copiée au préalable à l'aide du script Perl sur un Storage Element de la grille. Le programme BLAST nécessaire à la recherche de spécificité est également copié sur le Worker Node à partir de son LFN (Logical File Name) sur la grille. Pour une conception multiple, il est possible de soumettre au logiciel un fichier de la forme suivante:

```
Abiotrophia;25;0.8;50
Acanthopleuribacter;25;0.8;50
Acaricomes ;25 ;0.8 ;50
…
```

Un script Perl de supervision des jobs a été également développé. Il permet de resoumettre un job s'il échoue. La gestion de l'état du job se fait aussi par rapport au temps imparti de chaque job. En effet, dans la séquence consensus, nous avons remarqué que les extrémités 5' et 3' présentent souvent des dégénérescences plus importantes que le reste de la séquence. Les jobs correspondant à cette partie durent souvent plus longtemps que ceux correspondants aux autres fragments. La durée totale d'une conception est souvent corrélée avec la durée des calculs associés à ces fragments de début et fin de séquence (Tableau 15). Par exemple, pour les genres *Tetrapisispora* et *Thamnidium*, sur les cinq fragments de la séquence consensus, nous avons obtenu les temps de calcul suivants :

Tableau 15: Exemple de temps d'exécution des jobs de conception.

Job	1	2	3	4	5
Tetrapisispora	57m22s	7m2s	6m48s	6m26s	6m26s
Thamnidium	46m55s	5m58s	7m38s	6m	12m43

Nous présentons ici la partie centrale de la supervision des jobs sur la grille de calcul en se basant sur les interfaces données par le middleware gLite et les composants LCG et LFC.

```
#############################################################################
# Jobs Monitoring
#############################################################################
do
{

        $finished_jobs = 0;
        print "- - - - - - - - - - - - - - - - - - - - - - - - - - -\n";
        for ($debut=0; $debut<=$longueur-$lol; $debut+=$l)
        {
           if ($debut==0)
           {
              $debut2=0;
           }
           else
           {
              $debut2=$debut-1;
           }
           $nomjob="j_".$jobid."_$debut2";
REJOBS:
           $status = &j_status($debut2);
           if ($status eq "Aborted" || $status eq "Done(Failed)"
               || $status eq "Done(Exit Code !=0)"
               || $status eq "Cancelled")
           {
                #Soumission des jobs
                system ("rm -f $nomjob.job_id");
                system("glite-wms-job-submit -a
                        -o $nomjob.job_id $nomjob.jdl > $nomjob.submit");
                open(SUBMISSION,"$nomjob.submit");

                while(my $lines = <SUBMISSION>)
                {
                     if($lines =~ /[\w|\W]*Error[\w|\W]*/)
                     {
                           system ("rm -f $nomjob.job_id");
                           print "Submission Error occured....
                                  Resubmitting Job\n";
                           redo REJOBS;
                     }
                }
                close(SUBMISSION);
           }
           elsif ($status eq "Done(Success)" )
           {
                print "$jobid \t $finished_jobs
                       $jobs_list{$nomjob} \t--$status--\n";
                $finished_jobs++;
           }
           else
           {
                  print "$jobs_list{$count} \t--$status--\n";
                  print "Error in status\n";
           }
      }
}while($finished_jobs < $nbjob);
```

La parallélisation se fait donc en fragmentant la séquence consensus en plusieurs sous séquences et en lançant plusieurs conception simultanément. La durée moyenne d'une conception de sondes pour un genre est de quelques heures. L'architecture globale permettant de profiter de la grille de calcul est donnée par la Figure 63 et correspond à l'implémentation sur une interface utilisateur (UI) de la grille EGEE. Nous avons au total 3348 genres dont il faut faire la conception de sondes. La base de données comporte donc comme décrit plus haut les alignements finaux ainsi que les deux bases de données procaryotes et champignons. Quant à la base de données MySQL, il s'agit de la base qui contient les informations de conception et de tâches soumises. L'interface utilisateur UI de la grille est la plateforme sur laquelle les serveurs de base de données et le serveur web sont installés. La grille est utilisée de manière complètement transparente pour l'utilisateur et n'implique aucune connaissance de sa part des technologies employées. Enfin, pour une meilleure visibilité des données chaque utilisateur admet sa propre liste de conceptions. Les calculs ont été testés en utilisant un compte prédéfini. La liste des conceptions a été générée automatiquement et comporte tous les genres avec les paramètres choisis (longueur 25 ; threshold 0.8 ; nombre maximum d'hybridations croisées 50).

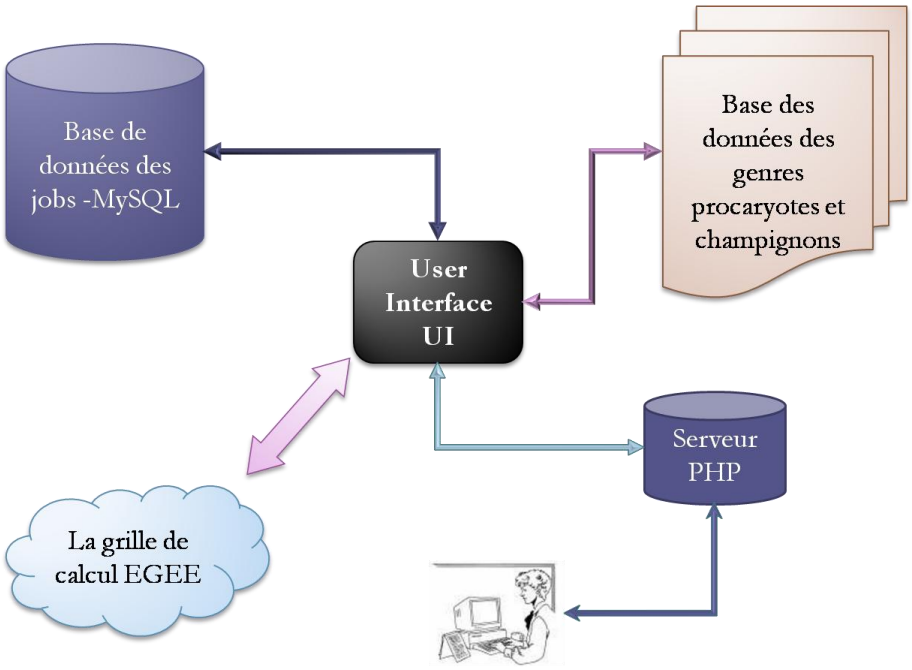


Figure 63: Architecture de l'application de conception de sondes pour biopuces phylogénétiques sur grille de calcul.

Dans un premier temps, nous avons effectué le calcul pour les champignons qui comptent 1441 genres différents avec les paramètres par défaut. La parallélisation étant faite uniquement sur la fragmentation des séquences consensus. Les conceptions ont été alors lancées par paquet de 20. Chaque paquet contient 5 jobs de 3 à 9 heures chacun. Nous avons estimé le temps de

calcul total pour les champignons sur une seule CPU à un an et 25 jours en prenant une durée moyenne de 6 heures et demi par conception. Les résultats sont donnés dans la partie suivante.

2.3.3. Résultats de la parallélisation sur grille de calcul

Après avoir lancé les calculs pour tous les genres champignons avec les paramètres par défaut en utilisant une approche par paquet de 20 conceptions à la fois, nous avons obtenu les résultats de la Figure 64. Seuls les jobs terminés avec succès dont les résultats finaux ont été obtenus sont comptabilisés dans le nombre de jobs Terminés avec succès. Si un job se termine avec succès mais aucun fichier de sortie n'a été produit, alors le job n'est pas considéré et il est resoumis à la grille en utilisant l'interface utilisateur. Après 7 semaines, nous avons obtenu les résultats pour 1356 genres des champignons ce qui représente environ 95% des données. Les 5% restants ont été resoumis dans un deuxième temps en les sélectionnant à partir de l'interface. Nous remarquons que la courbe est parabolique ce qui implique une meilleure efficacité de la grille pendant les premières semaines et une chute pendant les dernière semaines. Cette chute est due vraisemblablement à un certain nombre de jobs qui n'ont pas pu être exécutés.

Figure 64: Résultats d'exécution de 1356 genres champignons parmi 1441 genres sur la grille EGEE après 7 semaines.

Il faut également noter qu'il a souvent fallu gérer les pannes durant l'exécution des conceptions incluant l'indisponibilité de certaines ressources de la grille comme un CE[1] ou un SE[2]. Il peut s'agir également d'un blocage d'un job au niveau d'un CE. En effet, nous avons remarqué qu'il est possible que la ressource lui soit affectée mais qu'il ne s'exécute jamais. Aucun

outil n'a été développé au niveau du middleware pour gérer ce genre d'incidents. Une explication plausible peut venir du BDII[3] qui n'aurait pas mis à jour la liste des ressources disponibles. C'est pourquoi nous avons proposé une surcouche logicielle de supervision des jobs. L'outil est entièrement intégré dans l'application de conception de sondes et est complètement transparent pour l'utilisateur. En revanche, lorsqu'une tâche est perdue à cause de la panne d'un serveur (WMS, UI, Switch réseau) ou à cause d'une coupure de courant, les conceptions peuvent être relancées directement à partir de l'interface après une intervention sur le système. De plus, l'expiration du certificat de la grille peut engendrer la perte d'une tâche. Un démon logiciel assure la pérennité du certificat installé sur l'UI. Seul un redémarrage du système peut engendrer la perte massive de tâches. Nous envisageons de déployer tous les calculs des procaryotes (1907 genres) sur la grille en utilisant une fragmentation du consensus sur seulement 2 fragments ce qui donne un total de 3814 jobs.

2.4. Synthèse

La conception de sonde à ADN pour biopuce phylogénétique demande une attention particulière car les séquences manipulées sont des séquences hautement conservées entre microorganismes du même genre ou de la même famille. Il est donc primordial de proposer un outil adapté à cette problématique. Nous proposons ici un algorithme permettant de concevoir des oligonucléotides à la fois spécifique et exploratoire (découverte d'éventuelles nouvelles espèces) pour PhyloChip. De plus, ce logiciel profite de la puissance de calcul offerte par la grille européenne EGEE pour la parallélisation des conceptions à grande échelle. Le gain en performance dépend fortement de la disponibilité de la grille de calcul et montre qu'il est nécessaire de travailler plus sur la fiabilité de la grille de calcul afin d'optimiser encore la conception de sondes pour des biopuces phylogénétiques à grande échelle.

Dans la partie suivante, nous présenterons par ailleurs un nouvel algorithme de sélection de sonde ADN pour biopuces fonctionnelles. Nous verrons que cette approche partage des points en commun avec les biopuces phylogénétiques (la dégénérescence, la recherche de similarité) mais diffèrent sur plusieurs autres points (construction de la base de référence, les paramètres de sélections). Ce nouvel algorithme est déployé quant à lui sur une machine SMP avant d'être migré vers une ferme de calcul ou vers une grille de calcul.

[1]CE: Computing Element
[2]SE: Storage Element
[3]BDII: Berkely Database Information Index

3. HiSpOD : Logiciel de conception de sondes ADN pour biopuces métaboliques

Dans cette partie, un nouveau logiciel de conception de sondes ADN pour biopuces métaboliques sera décrit. Il a été développé en se basant sur la littérature afin de proposer une solution la plus complète possible dans l'état de nos connaissances pour la sélection de sondes spécifiques et sensibles pour tout type de biopuces et en particulier pour les biopuces métaboliques. Ce logiciel, appelé HiSpOD, offre une large gamme d'options et un large choix dans les critères de conception *in silico* pour une meilleure flexibilité par rapport aux différents besoins des utilisateurs. Il utilise la base de données ProtBase (III.2.2) comme base de référence de recherche d'hybridations croisées. Enfin, il est capable de déterminer des sondes dégénérées et spécifiques à partir de séquences ADN ou protéiques.

3.1. Principe de l'approche

HiSpOD est un logiciel de sélection d'oligonucléotides pour biopuces à ADN développé pour améliorer la détermination des sondes réalisée à l'heure actuelle par peu de logiciels (OligoWiz, ProbeMaker et OligoArray). Notre logiciel utilise une base de référence généraliste (ProtBase) comportant des données traitées représentant toutes les protéines connues correspondants aux séquences annotées, et leurs séquences CDSs présentes dans la banque de données EMBL et appartenant aux divisions procaryotes, champignons et environnement. Il permet de tester pour une séquence nucléique donnée dégénérée ou spécifique ou pour une séquence protéique, parmi tous les oligonucléotides possibles, ceux entraînant le moins d'hybridations croisées suivant des critères bien précis qui seront décrits par la suite. La base utilisée pour le test de spécificité BLAST est générée avec un outil d'interrogation et de construction de base de données FASTA suivant le souhait de l'utilisateur.

3.1.1. Génération des sondes

HiSpOD accepte en entrée une séquence au format FASTA. Pour l'exemple suivant, une séquence codant la sous-unité C de la MMOc (méthane-mono-oxygénase) intervenant dans les voies métaboliques du méthane est utilisée.

```
>gi|245213:412-1434 MMO gene cluster: mmoZ=gamma subunit of Protein
A...mmoC=Protein C [Methylosinus trichosporium, OB3b, Genomic, 3 genes,
1620 nt]
ATGTACCAGATCGTCATCGAGACCGAGGACGGAGAAACCTGTCGTCGAATGCGGCCCAGCGAGGATTGGA
TCTCGCGGGCTGAGGCAGAGCGTAATCTGCTCGCCTCCTGCCGAGCCGGCTGCGCCACCTGCAAGGCCGA
TTGCACGGACGGCGATTATGAGCTGATCGATGTGAAGGTCCAGGCCGTGCCGCCGGACGAGGAGGAGGAC
GGCAAGGTTCTGCTGTGCCGCACCTTTCCGCGCAGCGATCTGCATCTCCTCGTGCCTTACACCTATGATC
GCATCTCCTTCGAGGCGATTCAGACCAATTGGCTCGCCGAGATCCTCGCCTGTGATCGCGTGTCGTCCAA
TGTCGTGCGTCTCGTGCTGCAGCGCTCACGGCCGATGGCGGCGCGCATCTCGCTCAATTTCGTTCCCGGC
CAATTCGTCGACATCGAGATACCGGGCACGCATACACGGCGCTCCTACTCCATGGCCTCGGTCGCGGAGG
ATGGGCAGCTCGAATTCATCATCCGTCTGCTGCCGGACGGCGCCTTCTCGAAATTCCTGCAAACGGAAGC
GAAGGTCGGCATGCGCGTCGATCTGCGCGGACCGGCGGGCTCGTTCTTCCTGCATGATCACGGCGGCAGA
TCGCGCGTGTTCGTCGCCGGCGGCACGGGATTGTCGCCGGTGCTGTCGATGATCCGCCAGCTCGGCAAGG
CGAGCGATCCGTCGCCGGCGACGCTTCTGTTCGGCGTCACCAATCGCGAGGAATTGTTCTATGTCGACGA
GCTGAAGACTCTCGCGCAATCCATGCCGACCCTCGGCGTGCGCATAGCGGTGGTCAATGACGACGGCGGC
AATGGCGTCGACAAGGGAACGGTGATTGATCTTCTGCGGGCCGAGCTCGAGATAGACTTGCTCCTTGGGC
ACGCCCGCCGTCGCCGCCGCCGCGAAACGGCTCGATCATGCCGGGAGGACCACAGAGATAGATGTCCGGC
TTGGCGTTCGGACTTTCTCGAGAAATTCCTGGCGAGCGGCTGA
```

Il s'agit de générer à partir de la séquence d'entrée de longueur L, toutes les sous-séquences d'une longueur L_0. Ces sous séquences vont correspondre en réalité à un fichier où un ou plusieurs oligonucléotides spécifiques seront stockés. Le nombre absolu de fichiers créé est alors égal à $N = L - L_0 + 1$. Mais le nombre final de fichiers créés est $N_f \leq N$. Dans l'exemple, nous avons une séquence de longueur 1023 pb. *HiSpOD* est capable de générer jusqu'à 1000 fichiers d'oligonucléotides de 24 mers pour cette séquence. On obtient alors les fichiers suivants :

mmoC_ATGTACCAGATCGTCATCGAGACC_1_24.fasta

```
>1
ATGTACCAGATCGTCATCGAGACC
```

mmoC_TGTACCAGATCGTCATCGAGACCG_2_25.fasta

```
>1
TGTACCAGATCGTCATCGAGACCG
```

mmoC_GTACCAGATCGTCATCGAGACCGA_3_26.fasta

```
>1
GTACCAGATCGTCATCGAGACCGA
```

…

mmoC_GAGAAATTCCTGGCGAGCGGCTGA_1000_1023.fasta

```
>1
GAGAAATTCCTGGCGAGCGGCTGA
```

Comme indiqué précédemment, il est possible de soumettre une séquence dégénérée issue directement d'un alignement multiple à l'aide d'outils tels que *ConsensusMaker* [1], *Edtaln* [2], ou *MASH* (Chappey et al., 1991). Cela permet de ne plus cibler un microorganisme en particulier mais de cibler tous les microorganismes ayant potentiellement l'activité métabolique d'intérêt. A

[1]ConsensusMaker: http://www.hiv.lanl.gov/content/sequence/CONSENSUS/consensus.html
[2]Edtaln: http://bioinfo.hku.hk/services/analyseq/cgi-bin/edtaln_in.pl

partir d'une séquence protéique ou d'une séquence consensus, la traduction inverse d'oligopeptide est effectuée avec le programme présenté dans la partie III.2. Après la génération de tous les oligonucléotides, les tests de spécificité sont initiés. Il est également possible de donner à l'entrée du programme une séquence générée à partir de l'un des logiciels de génération de séquences consensus.

Que ce soit pour les séquences nucléiques dégénérées ou pour les séquences protéiques de références, HiSpOD génère pour chaque sous-séquence de longueur L_0 toutes les possibilités d'oligonucléotides. Par exemple, pour le premier oligopeptide « MAATTESV » de longueur L_0 = 8 aa (équivalent à des oligonucléotides de 24 mers), on obtient 1×4×4×4×4×2×6×4=12288 oligonucléotides possibles d'après notre algorithme de traduction inverse et sont enregistrés dans un fichier « **mmoC_MAATTESV_1_8.fasta** ».

3.1.2. Vérification des critères intrinsèques des sondes

Dans la conception d'oligonucléotides pour biopuces à ADN, certains critères de sélection sont intrinsèques et propres à la séquence oligonucléotidique et dépendent donc uniquement de sa composition. HiSpOD offre la possibilité de déterminer la longueur, la température de fusion minimale, la température de fusion maximale (qui sont calculée à partir de l'oligonucléotide en utilisant la méthode d'ajustement II.1.2.3), le pourcentage de GC, la complexité (en termes de nombre maximal de bases consécutives identiques « stretch ») et la dégénérescence maximale autorisée pour un oligonucléotide dégénéré ou pour un oligopeptide. Un oligonucléotide dégénéré (respectivement un oligopeptide) qui ne vérifie pas ces critères n'est pas retenu par la suite. Si au moins un des oligonucléotides générés pour une position donnée ne vérifie pas ces critères, alors l'oligonucléotide (respectivement l'oligopeptide) est supprimé et le fichier correspondant est supprimé. Par exemple, pour le fichier suivant :

unknown_ ATGTACCAGATCGTCATCGAGACY_1_24.fasta

```
>1
ATGTACCAGATCGTCATCGAGACC
>2
ATGTACCAGATCGTCATCGAGACT
```

Nous avons les propriétés suivantes :

- Dégénérescence = 2
- Oligo N° 1
  - Longueur = 24
  - Pourcentage GC = 50 %
  - Température de fusion = 53.71
  - Complexité = 2 Stretch de 2 bases C
- Oligo N° 2
  - Longueur = 24
  - Pourcentage GC = 45.8 %
  - Température de fusion = 50.80
  - Complexité = 1 Stretch de 2 bases C

Si on avait fixé le pourcentage GC à 50%, seul l'oligo N°1 serait gardé. On obtient alors le fichier suivant après cette étape. Cet oligonucléotide dégénéré ne sera pas sélectionné car il ne cible pas toutes les séquences possibles et par conséquent omet certaines espèces.

unknown_ ATGTACCAGATCGTCATCGAGACY_1_24.fasta

```
>1
ATGTACCAGATCGTCATCGAGACC
```

3.1.3. Test de spécificité par recherche de similarité

Comme évoqué précédemment, la base de données utilisée pour HiSpOD est construite à partir de la base de données relationnelle ProtBase. Elle est formatée et fragmentée à l'aide de mpi-formatdb, du package MPI-BLAST et est installée sur une machine SMP de 16 coeurs (Figure 65). L'intérêt est de profiter de la puissance de calcul pour obtenir des meilleurs temps de conception puisque la recherche de similarité est l'étape la plus couteuse en termes de temps de calcul. En fait, chaque fichier obtenu après l'étape de vérification des critères intrinsèques est comparé contre la base de données. Nous avons vérifié que l'utilisation d'une base fragmentée avec le maximum de fragments augmente la performance du BLAST (Missaoui, 2006), la meilleure performance étant atteinte pour une valeur égale à N+1 coeurs si N est le nombre de fragments de base. Sachant que les performances pour $x \leq 16$ processeurs et 15 fragments sont toujours meilleures que pour $y < x$ et 15 fragments, nous avons choisi de fragmenter entièrement la base sur 15 fragments et d'utiliser 16 cœurs pour obtenir les meilleures performances possibles.

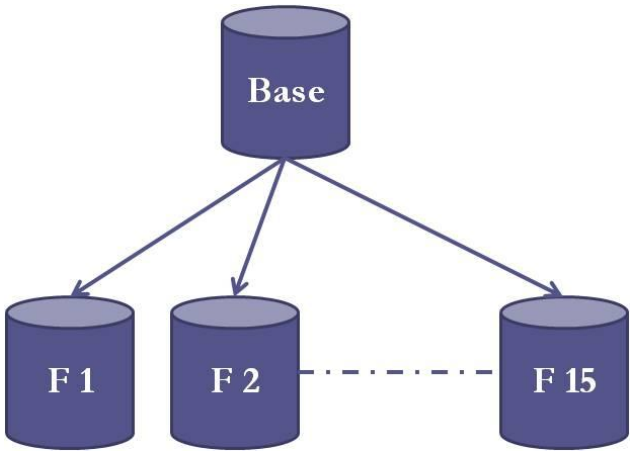


Figure 65: Fragmentation de la base de données pour HiSpOD sur une machine de 16 cœurs.

Chaque oligonucléotide comparé donne un fichier de sortie BLAST. Nous traitons la sortie pour en extraire les séquences qui présentent potentiellement des hybridations croisées afin de vérifier les critères de Kane à savoir le pourcentage de similarité et le nombre de nucléotides consécutifs identiques.

unknown_ ATGTACCAGATCGTCATCGAGACY_1_24.fasta.blast

```
……
 Identities = 24/24 (100%), Gaps = 0/24 (0%)
 Strand=Plus/Plus
Query  1    ATGTACCAGATCGTCATCGAGACC 24
            ||||||||||||||||||||||||
Sbjct  653  ATGTACCAGATCGTCATCGAGACC 672
……
```

3.1.4. Vérification des hybridations croisées

L'étape suivante consiste à parcourir le fichier de sortie BLAST et de construire une liste non redondante des identifiants de séquences qui croisent potentiellement avec les oligonucléotides d'un même fichier d'après les critères suivants fixés par l'utilisateur :

- Le pourcentage d'identité avec l'oligonucléotide testé
- Le nombre de base consécutive identique « stretch » de similarité

Nous obtenons alors un fichier contenant les identifiants des séquences dans la base de données *ProtBase*. Ce fichier est un fichier intermédiaire contenant les identifiant des séquences dans la base de données. Il est traité dans les étapes suivantes.

unknown_ ATGTACCAGATCGTCATCGAGACY_1_24.fasta.blast.cross

```
12454
6546
45412
466
21546
64667
354
12164313
144445
…
```

3.1.5. Transformation de la sortie BLAST et création des groupes d'hybridations croisées

La liste obtenue pour chaque fichier est transformée ensuite en un fichier au format FASTA en interrogeant la base de données relationnelle ProtBase pour en extraire les séquences nucléiques référencée par les identifiants précédemment extraits afin de définir les hybridations croisées potentielles. HiSpOD traite alors cette sortie pour créer un fichier d'hybridation croisée en regroupant à l'aide du programme BlastClust les séquences suivant deux critères fixés par l'utilisateur :

- Le pourcentage d'identité relatif
- Le pourcentage de la longueur prise en compte

Ces deux paramètres permettent d'ajuster les groupes créés afin de regrouper ensemble les séquences qui présentent des similarités fortes pour factoriser les hybridations croisées fictives en hybridations croisées réelles. Le résultat final est donné sous forme de fichier résultat par sonde sélectionnée. De manière générale, les séquences identiques à la séquence de référence sont regroupées ensemble indiquant qu'il ne s'agit pas d'une hybridation croisée, tandis que les autres séquences sont regroupées par groupe homogène du point de vu de la similarité. On obtient alors un fichier de sortie comme le suivant. Les groupes sont identifiés par leur couleur dans cet exemple. Mais en réalité, chaque ligne contient un groupe, les séquences d'un même groupe sont séparées par un espace. Il s'agit d'un affichage qui sera amélioré ultérieurement.

unknown_ ATGTACCAGATCGTCATCGAGACY_1_24.fasta.blast.cross.clean

```
Q8RQD4:1357:Pseudomonas_putid:4_hydrox
B238971_77_p:1348:Pseudomonas_putid:4_h
Q9S152:1346:Comamonas_testost:4_hydrox
P51018:1285:Pseudomonas_putid:4_hydrox
P51018:1282:Pseudomonas_putid:pyruvate
Q9EVJ1:1282:Pseudomonas_putid:4_hydrox
Q9EVJ1:1282:Pseudomonas_putid:4_hydrox
Q9RA00:1282:Pseudomonas_putid:4_hydrox
Q4J3S2:1201:Azotobacter_vinel:HMG_CoA_
AB238971_78_p:1144:Pseudomonas_putid:4_o
```

```
Q8RQD3:1144:Pseudomonas_putid:4_olaloc
Q4J2W6:1113:Azotobacter_vinel:HMG_CoA_
Q4IUD1:1105:Azotobacter_vinel:HMG_CoA_

EP287753_2_p:1077:marine_metagenome:hypo
EP287283_0_p:1058:marine_metagenome:hypo
EP287283_2_p:1026:marine_metagenome:hypo
EP287753_1_p:784:marine_metagenome:hypot
EP617583_2_p:782:marine_metagenome:hypot
EM702417_0_p:588:marine_metagenome:hypot
EM542674_1_p:338:marine_metagenome:hypot

Q2VLC5:3153:Burkholderia_cepa:BphF Q84EP5:1367:Cupriavidus_oxala:putative
Q120N9:1359:Polaromonas_sp__J:pyruvate
…
```

Chaque ligne du fichier résultats correspond à un groupe d'hybridation croisée. En d'autres termes, une hybridation croisée est considérée lorsqu'on détermine un ensemble de séquences fortement homogènes et qui croisent systématiquement avec la sonde déterminée. La sortie est suffisamment détaillée pour retrouver exactement les séquences qui hybrident potentiellement avec la sonde. Chaque séquence qui croise potentiellement est identifiée par son numéro d'accession

3.2. Architecture du logiciel

HiSpOD est un logiciel conçu et développé dans l'objectif d'exploiter un système d'information pour la centralisation des données génomiques et utilise une base de données relationnelles (*ProtBase*) afin de simplifier au mieux la conception de sonde à ADN. Il exploite une architecture parallèle (SMP) de 16 cœurs (8 Core-Duo de 1.8 GHz) et de 128 Go de RAM avec une capacité de stockage de 2 To. La base de données est installée sur un SGBD Oracle 10g version 10.2. Le logiciel est développé en Perl version 5.10 et utilise les modules bioinformatiques de BioPerl version 1.5.2. Il utilise la version 1.4.0 de MPI-BLAST et la version de 2.2.20 de BalstClust.

HiSpOD utilise notre programme de traduction inverse pour la conception de sondes dégénérées et des sondes spécifiques de régions protéiques comme nous l'avons indiqué au début de cette partie. Il permet donc un large panel de conception de sondes pour des biopuces fonctionnelles et métaboliques. L'un de ses points forts est sa base de données de références pour la recherche de spécificité (ProtBase) qui englobe la totalité des séquences génomiques connues pour les procaryotes, les champignons et les séquences environnementales. Le schéma de la Figure 66 récapitule les étapes de la conception avec le logiciel HiSpOD.

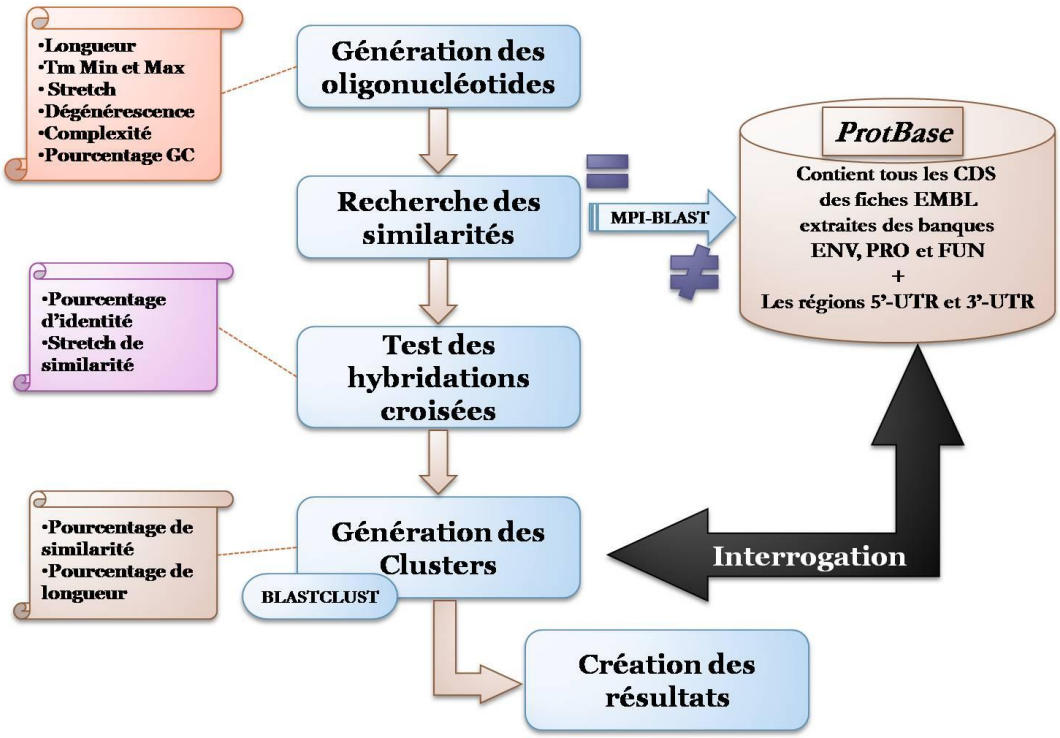


Figure 66: Les étapes principales de la conception de sondes ADN avec HiSpOD.

L'étape de reconstruction des résultats demande l'interrogation de la base de données pour le regroupement des hybridations croisées potentielles. Cette étape est cruciale car elle détermine pour une sonde donnée le nombre possible d'hybridations croisées dans le mélange cible. L'affichage de résultats est d'une forme textuelle. Une interface graphique est en cours de développement.

3.3. Résultats et performances

Nous avons réalisé au laboratoire des conceptions d'oligonucléotides à l'aide de HiSpOD. Ici, nous présentons les résultats pour les gènes mmoC (métane mono-oxygénase), tceA (trichloroethene reductive dehalogenase sous-unité A) et benzA (dégradation du benzene). Nous avons conçu des oligonucléotides spécifiques de 20, 25, 40 et 50 mers pour chacun des gènes. Nous avons fixé la complexité à 10 nucléotides consécutifs, la e-value BLAST à 1000, le stretch à 15, le pourcentage de similarité BlastClust à 90 et la longueur prise en compte à 0 ce qui signifie un regroupement quelque soit la longueur des séquences obtenues présentant une hybridation croisée. Les autres paramètres ainsi que les temps de calculs sont présentés dans le Tableau 16.

Tableau 16: Récapitulatifs des critères de conception de sondes réalisées pour les séquences de références des 3 gènes *mmoC*, *tceA* et *benzA*.

mmoC							
Longueur	**T_m min**	**T_m max**	**%id**	**Temps (minutes)**	**Nombre d'oligos potentiels**	**Nombre d'oligos vérifiant les critères**	**Nombre final d'oligos sélectionnés**
20	39	55	90	101	1004	479	4
25	47	63	85	69	999	297	4
40	59	75	75	68	984	267	6
50	64	79	75	65	974	239	5

tceA							
Longueur	**T_m min**	**T_m max**	**%id**	**Temps (minutes)**	**Nombre d'oligos potentiels**	**Nombre d'oligos vérifiant les critères**	**Nombre final d'oligos sélectionnés**
20	39	55	90	243	1646	1137	14
25	47	63	85	239	1641	1042	11
40	59	75	75	327	1626	1283	5
50	64	79	75	344	1616	1264	5

benzB							
Longueur	**T_m min**	**T_m max**	**%id**	**Temps (minutes)**	**Nombre d'oligos potentiels**	**Nombre d'oligos vérifiant les critères**	**Nombre final d'oligos sélectionnés**
20	39	55	90	173	545	334	5
25	47	63	85	122	540	205	3
40	59	75	75	130	425	216	3
50	64	79	75	142	515	222	4

Sachant que la séquence *mmoC* est d'une longueur de 1052 pb, tceA est d'une longueur de 1712 pb et *benzA* est d'une longueur de 1392 pb, et que le nombre de cœurs utilisées pour la recherche de similarité BLAST était égale à 6 dans un premier temps, nous remarquons que la longueur des sondes sélectionnées n'a pas d'incidence directe sur la performance du logiciel (Figure 67). En revanche, pour les mêmes critères de sélection, le temps de calcul augmente lorsque la longueur totale de la séquence augmente. Mais ce sont les critères de sélections qui déterminent principalement le temps de calcul général. En effet, le pourcentage de GC par exemple qui détermine implicitement la température de fusion de l'oligonucléotide influe beaucoup sur le nombre de sondes sélectionnées et par conséquent sur le temps de calcul.

Figure 67: Exemple de temps d'exécution de HiSpOD pour trois gènes.

Comme nous l'avons évoqué auparavant, l'une des étapes consommatrice de ressources est la comparaison BLAST des oligonucléotides contre la base de données. Une utilisation optimale du logiciel passe par une utilisation parfaite des ressources disponibles avec BLAST sur notre architecture SMP de 16 processeurs. En effet, nous avons réalisé des tests de performance sur un fichier d'oligonucléotides : l'un contient une seule séquence (oligonucléotide spécifique) et l'autre contient 200 séquences (oligonucléotide dégénéré) afin de reproduire les conditions réelles d'exécution de HiSpOD. Les résultats sont donnés dans le Tableau 17 et montrent que lorsqu'on fragmente de manière optimale la base de données de références – c'est-à-dire sur 15 fragments – la performance optimale est donnée comme prévue pour N+1 = 16 cœurs utilisées où N=15. Les options du programme BLAST adaptées à la recherche de similarité pour des oligonucléotides sont les suivantes:

-e = 1000 (e-value)

-F = F (filtre de complexité non activé)

-S = 1 (test de similarité sur un seul brin)

-W = 7 (taille du mot)

-b = 500 (Nombre de séquences de la base à afficher)

-v = 500 (Nombre de séquences dont on affiche la ligne de commentaire)

Tableau 17: Temps de calculs pour une séquence oligonucléotidique spécifique de 18 mers et une séquence oligonucléotidique dégénérée de 18 mers.

1 séquence = 44.85s en local					
Nombre CPUs	**3**	**8**	**16**	**32**	**64**
Temps (secondes)	43.98	9.27	6.52	9.20	10.96
Speed-up	1.01	4.81	6.83	4.84	4.08
2 séquence = 42.61s en local					
Nombre CPUs					
Temps (secondes)	71.01	12.80	8.05	12.15	12.44
Speed-up	0.61	3.35	5.34	3.53	3.45

Comme attendu, nous avons obtenu les meilleures performances pour une utilisation optimale des cœurs disponibles sur le SMP de 16 cœurs. Afin de vérifier que les performances ne peuvent pas augmenter en surchargeant le système, nous avons lancé des simulations pour 32 et 64 tâches en parallèle pour simuler 2 ou 4 utilisateurs. Dans ce cas de figure, les performances ne chutent que très peu. La Figure 68 montre le comportement de MPI-BLAST pour le cas d'utilisation de notre application. Le meilleur speed-up est obtenu donc pour 16 cœurs pour lesquels la meilleure performance est obtenue sachant que chaque tâche utilise un cœur.

Nous avons constaté également que les paramètres de BLAST influent de manière considérable sur le temps de calcul total. Par exemple, l'option –S permet de préciser si la recherche d'alignement se fait sur les deux brins ou sur un seul brin. La durée de recherche peut donc passer du simple au double. De plus il est possible de préciser le nombre de données à afficher. Si nous prenons en compte la durée d'écriture dans le fichier résultat, il est évident que la durée d'écriture de quelques séquences diffère de l'écriture de plusieurs centaines de séquences. Cette différence est accentuée par l'augmentation du nombre de séquence dans le fichier des sondes. Les options –v et –b calibrent souvent la taille du fichier de sortie en les exploitant avec la e-value.

Figure 68: Comportement de MPI-BLAST pour une séquence oligonucléotidique spécifique et une séquence oligonucléotidique de dégénérescence 2 sur une SMP de 16 cœurs en utilisant 15 fragments de base.

L'un des points forts de notre logiciel HiSpOD est sa capacité à utiliser une architecture parallèle telle que la ferme de calcul locale du LIMOS. Il est donc possible de réaliser simultanément plusieurs conceptions pour des gènes différents. De plus en le comparant à d'autres logiciels tels qu'OligoWiz, ProbeMaker et OligoArray qui ont les mêmes logiques d'utilisation pour la conception de sondes à ADN, nous pouvons décrire les particularités de chaque approche dans le Tableau 18.

Tableau 18: Comparaison des paramètres de HiSpOD avec trois autres logiciels.

Logiciel	Tm	%GC	Complexité	Similarité	Stretch	Structure Secondaire	Dégénérescence
OligoWiz	•			•			
OligoArray	•	•	•		•		
ProbeMaker	•	•	•			•	
HiSpOD	•	•	•	•	•	•	•

Les autres paramètres comme la longueur ou les critères d'hybridations croisées (la similarité et le stretch) sont souvent des critères prédéfinis dans la plupart des logiciels. HiSpOD intègre la majorité des critères de sélection de sonde en comparaison avec les autres logiciels. Il

permet en plus de concevoir des sondes pour des séquences dégénérées et donne à l'utilisateur la possibilité de fixer tous les paramètres, lui donnant ainsi une meilleure flexibilité d'utilisation. Nous envisageons de migrer l'application sur la grille locale du LIMOS présentée précédemment et qui dispose de 272 cœurs. Nous avons fait une étude de performance générale en prenant en compte le cas de sondes fortement dégénérées (jusqu'à 1000 oligonucléotides). Nous avons également fragmenté la base de données pour passer à l'échelle conseillée par Lin et al., (2008) c'est-à-dire 128 fragments de base. Les tests de performances sont très prometteurs (Figure 69) pour l'utilisation à une plus grande échelle de notre application et sa mise à disposition pour la communauté scientifique.

Figure 69: Performance de MPI-BLAST pour des oligonucléotides fortement dégénérées sur la ferme de calcul du LIMOS en utilisant 128 fragments de base et un nombre croissant de cœurs.

Comme nous l'avons vu dans la partie III, MPI-BLAST est plus performant lorsqu'on travaille sur une échelle de 20 000 séquences par exemple. Plusieurs conceptions à l'aide de HiSpOD peuvent être lancées simultanément pour profiter au maximum de la puissance de la ferme de calcul locale. Comme présenté précédemment, le meilleur résultat est obtenu avec 129 cœurs pour 128 fragments de base. Nous avons en effet adapté le programme MPI-BLAST à notre logiciel pour gagner du temps de calcul vu qu'il est installé sur une architecture parallèle avec la librairie MPI.

3.4. Synthèse

La conception de sonde pour biopuces métaboliques est un défi qui demande une méthodologie précise pour le choix des paramètres de détermination des oligonucléotides et suffisamment de puissance de calcul pour l'obtention rapide des résultats. HiSpOD est un logiciel entièrement développé au laboratoire qui répond parfaitement à ces exigences en proposant une flexibilité au niveau du type de sondes à concevoir (dégénérées ou spécifiques pour des gènes ou des protéines). Il intègre la plupart des critères de sélection de sondes retrouvés dans la littérature et qui sont connus pour intervenir efficacement dans l'obtention de sondes spécifiques. L'utilisation de BlastClust comme outil de regroupement des hybridations croisées permet de factoriser par taxonomie (genre ou famille) le nombre total d'hybridations aspécifiques théoriques. L'intégration également d'un outil de traduction inverse de séquence dégénérées (protéiques ou nucléiques) permet d'apporter un aspect exploratoire aux sondes sélectionnées pas le logiciel. Notre logiciel profite d'une architecture parallèle. Il est donc possible de lancer simultanément plusieurs conceptions de sondes. L'utilisation d'architectures parallèles telles que les SMPs ou les ferme de calcul permet une augmentation significative des performances du logiciel.

4. Conclusion

Nous avons présenté, dans ce chapitre, deux logiciels de conception de sondes pour biopuces à ADN dans le contexte de l'écologie microbienne et la découverte de la biodiversité. Ces deux algorithmes sont capables d'utiliser des architectures à haute performance comme les grilles de calcul et les fermes de calcul.

D'une part, nous avons calculé à l'aide du logiciel de conception de sondes pour les puces phylogénétique les sondes potentielles de 1441 genres de champignons correspondant à plusieurs centaines de milliers d'oligonucléotides. Environ 95% des résultats, équivalent à 13 mois de calcul, ont été obtenus en 7 semaines grâce aux ressources offertes par la grille européenne EGEE. Les 5% restants ont été obtenus par la suite après avoir relancé les calculs. Nous avons alors réussi à proposer une base de données complète de sondes ADN pour les champignons. Les genres procaryotes ont été également construits. La base de données des sondes est en cours de calcul à l'aide de notre logiciel en utilisant la grille de calcul.

D'autre part, nous avons proposé une nouvelle base de données générale de séquences des microorganismes d'environnement. Elle permet de centraliser les données génomiques métaboliques des bactéries, des champignons et environnementales, qui sont répertoriés dans les banques de données internationales comme l'EMBL. Elle est utilisée comme base de références pour notre logiciel de conception de sondes pour biopuces à ADN en écologie microbienne. Nous avons aussi développé un nouvel algorithme de traduction inverse d'oligopeptides utile pour la conception de sonde spécifique de gènes codant pour une enzyme donnée notamment avec l'utilisation d'une démarche d'ingénierie dirigée par les modèles et particulièrement de méta-programmation.

Enfin nous avons proposé un nouveau logiciel de sélection de sonde appelé HiSpOD qui nous a permis de calculer des oligonucléotides de plusieurs centaines de séquences impliquées dans des voies métaboliques d'intérêt (plus précisément les voies de dégradation des solvants chlorés). Il permet d'exploiter l'intégration de plusieurs critères de sélection de sonde et profite d'une base de données généraliste lui permettant une recherche efficace des hybridations croisées potentielles.

Les architectures parallèles utilisées (Grille, SMP et ferme de calcul) ont été utiles pour augmenter les performances de nos algorithmes. En effet, nous avons démontré que chaque

problème est adapté à une architecture en particulier. Par exemple, pour une conception d'oligonucléotides à grande échelle avec une petite quantité de données transférée, la grille de calcul parait une solution intéressante. D'un autre côté, nous avons vu que la parallélisation sur la grille EGEE du programme BLAST peut poser des problèmes si la base de données de comparaison est d'une grande taille (environ 9 millions de séquences) et que le nombre de tâches est grand et leur durée courte (quelques minutes), mais s'avère intéressante lorsque la base est de petite taille (quelques mégaoctets) et que les tâches ont une durée longue (quelques heures). Nous avons vu également que les architectures de type SMP ou fermes de calcul peuvent augmenter la performance des logiciels de conception de sondes pour biopuces fonctionnelles. Il est donc nécessaire d'adapter un problème de conception de sondes à la taille des données et à l'architecture parallèle utilisée pour obtenir les meilleures performances. Le tableau ci-dessous synthétise ces différentes orientations en indiquant la meilleure option grâce au nombre croissant de points.

Type de Problème	**Taille de la base de données**	**Nombre de tâches longues (quelques heures)**	**Grille de calcul**	**SMP**	**Ferme de calcul**
Alignement local ou global	>~ 1 Go	Quelques dizaines	•	•••	••••
	>~ 1 Go	>Quelques centaines	•••	•	••
	Quelques Mo	-	•••	••••	••••
Alignement multiple	< quelques milliers	<Quelques centaines	?	••	••••
Conception de sondes pour biopuces phylogénétique	Quelques dizaines de Mo	Quelques milliers	••••	•	•
	Quelques dizaines de Mo	Quelques tâches	•	••••	•••
Conception de sondes pour biopuces fonctionnelles	>~ 1 Go		?	••	••••

Conclusions et perspectives

Contributions

La sélection de sondes pour biopuces à ADN est loin d'être une tâche triviale. En effet, la complexité des algorithmes de conception de sondes est due essentiellement à la difficulté de combiner une multitude de paramètres pour la recherche d'une sonde optimale (spécifique et sensible) ainsi qu'à la taille de données génomiques qu'il est nécessaire de traiter et qui est aujourd'hui toujours en croissance permanente. De plus, l'utilisation répétitive de certains logiciels bioinformatiques, notamment d'alignement de séquences, engendre souvent des temps de calcul prohibitifs. Le programme d'alignement local de séquences BLAST est utilisé pour la recherche de spécificité afin de prédire les éventuelles hybridations croisées. Le programme d'alignement multiple de séquences est utilisé pour déduire une séquence consensus représentative d'un taxon. L'utilisation de leurs versions parallèles (Darling et al., 2003 ; Li, 2003) dédiées au Cluster dans les logiciels de conception de sondes permet une augmentation de la performance globale comme nous l'avons présenté dans cette thèse.

De plus, l'utilisation d'architectures distribuées telles que les grilles de calcul pour un calcul à très grande échelle permet des gains de temps considérables. En effet, nous avons présenté un logiciel entièrement déployé sur la grille de calcul EGEE. Il permet de réaliser la conception de sondes de plusieurs centaines de séquences pour des biopuces phylogénétiques. Il est entièrement automatisé. Nous avons alors créé une banque de données spécifiques aux genres procaryotes et champignons utile pour la conception de sondes spécifiques complètes des microorganismes connues. L'approche permet à la fois de cibler ces genres ou des groupes taxonomiques supérieurs et de détecter des microorganismes inconnus dans les environnements étudiés. Une base de sondes spécifique a été créée à l'aide de ce logiciel. Les biopuces complètes permettront l'exploration d'environnements complexes.

Nous avons aussi proposé un nouvel algorithme générique appelé HiSpOD pour la conception de sondes pour biopuces fonctionnelles. Il permet, contrairement à la majorité des logiciels existants, d'intégrer la plupart des paramètres de sélection d'oligonucléotides comme, entre autres, la température de fusion, la complexité, la longueur ou la dégénérescence. Il peut s'appliquer à des sondes dégénérées ou spécifiques. Il utilise une base de données appelée ProtBase développée spécialement pour intégrer toutes les séquences microbiennes répertoriées à ce jour et mise à jour à l'aide d'un outil d'extraction adapté. Enfin ce dernier logiciel permet également la conception de sondes dégénérées qui cible spécifiquement des protéines d'intérêt ou des voies métaboliques. Il profite d'une architecture parallèle (machine multiprocesseurs) pour

effectuer la sélection de sondes afin de gagner en performance. Les premières biopuces sont en cours de test pour l'évaluation des capacités microbiennes de dégradation des solvants chlorés.

Pour la gestion de la dégénérescence du code génétique, nous avons développé des algorithmes de traduction inverse d'oligopeptides utiles dans la recherche de sondes qui ciblent des protéines. Nous avons fait appel à l'ingénierie dirigée par les modèles et notamment la transformation de modèles pour générer une version optimisée et parallèle de ces codes. De plus, l'utilisation d'un système d'information intégré a permis de simplifier considérablement la structuration des données.

La réalisation d'un programme de parallélisation du programme BLAST sur la grille de calcul EGEE nous a permis d'envisager de réaliser les recherches de similarité pour toutes les combinaisons possibles d'oligonucléotides pour les longueurs d'intérêt. En effet, même si ce programme a montré de bonnes performances sur des petites données, néanmoins, il reste impossible de garantir à 100% la totalité des résultats, comme nous l'avons vu, lorsque l'on travaille sur des données plus importantes (quelques jours à quelques mois de calcul/CPU).

De plus, la quantité de données transférée à travers le réseau de la grille EGEE est de quelques Téra Octets lorsqu'on lance des milliers de tâches. Ceci peut poser problème si les zones de stockages sont géographiquement éloignées des zones de calculs d'où l'intérêt de faire un réplica le plus proche possible pour chaque site de calcul. En appliquant cette approche avec des données moins importantes mais avec un nombre équivalent de tâches lors de la conception de sondes pour biopuces phylogénétiques, nous avons remarqué naturellement une meilleure fiabilité de la grille. Le middleware gLite de la grille permet une gestion de haut niveau des ressources informatiques au niveau d'une grille, mais des travaux considérables sont encore en cours (Han et Youn, 2009 ; Alef et al., 2009 ; Shiers, 2009). Pour augmenter la fiabilité de la grille certaines applications font plusieurs dizaines de réplicas des mêmes tâches en espérant qu'au moins une se termine correctement (Smith et al., 2009).

L'une des notions émergentes autour des grilles de calcul est ce qu'on appelle en anglais « dependability », notion qui d'après Grimshaw (2005) « *will include reliability in some cases, availability in others, data integrity and confidentiality (security) in essentially all services, and possibly safety in some circumstances.* ». On ne compte pas moins de 550 publications scientifiques en 2009 qui traite de la tolérance aux pannes pour l'ordonnancement des processus sur grille de calcul. Cela montre clairement le besoin en grilles plus robustes et plus sures. Dans les applications présentées dans cette thèse et qui utilisent la grille de calcul, nous avons proposé un algorithme de supervision des

tâches et de tolérance aux pannes, basé essentiellement sur les statuts des tâches et leurs traces en cours d'exécution en s'aidant des APIs (Interface de programmation d'applications) proposées par gLite. Il est indiscutable que les architectures HPC (calcul à haute performance) de type ferme de calcul, SMP (Symmetric Multiprocessor) sont plus fiables mais elles sont adaptées pour des problématiques de calcul à moyenne échelle.

Il est important de rappeler les problématiques liées à la sécurité au niveau de la grille. Là aussi, de nombreux travaux ont été réalisés (pas moins de 740 publications en 2009) surtout lorsqu'il s'agit de traiter des données biomédicales, militaires ou de l'ordre de la sécurité publique. De plus, plusieurs projets innovants ont été développés depuis quelques années pour proposer des plateformes de développement d'application de grille de calcul comme MyGrid, g-Eclipse, Taverna ou GrADS. Les applications basées sur les SOAs (Service Oriented Architecture) et les Web services sont étudiées également pour une utilisation au niveau des grilles de calcul dans le cadre des OGSA (Open Grid Service Architecture). Plusieurs projets de collaborations par domaines d'application ont vu le jour grâce à cette standardisation comme Embrace (bioinformatique), AccessGrid (multimedia et middleware) ou GridPP (physique des particules).

Les applications logicielles autour des puces à ADN sont fortement consommatrices de ressources de calcul et d'espace de stockage (Maglogiannis et al., 2007). Les grilles de calcul apportent une réponse à certains types de problèmes qui acceptent un certain degré de pannes. Les logiciels de conception de sondes à ADN sont parallélisables sur grille de calcul mais nous sommes encore confrontés à des compromis quant à la fiabilité de la grille si nous voulons travailler à grande échelle. Il est en effet souvent nécessaire d'accepter de resoumettre un nombre significatif de tâches lorsque l'on soumet plusieurs centaines de tâches.

Perspectives des travaux

Plusieurs applications de nos outils sont déjà en cours. En effet, nous validons la conception réalisée par le logiciel de détermination de sondes pour biopuces phylogénétiques sur les genres des champignons. Les calculs concernant les procaryotes sont en cours d'exécution sur la grille de calcul. L'objectif est de créer une biopuce phylogénétique complète des 3349 genres bactériens. Elle permettra d'étudier en profondeur les mécanismes d'adaptation de ces microorganismes quelque soit l'environnement considéré. Nous utilisons actuellement la grille de calcul EGEE pour la deuxième conception de sondes à ADN sur les genres procaryotes.

Nous avons par ailleurs constaté que notre logiciel peut être amélioré pour compenser le manque de fiabilité de la grille en ajoutant un module de resoumission automatique après l'échec d'un processus. Cela permettra de fiabiliser encore plus l'application. Du point de vue de la sécurité, il est primordial de proposer des solutions logicielles qui protègent les applications déployées sur la grille des attaques de personnes malintentionnées. Enfin, la sortie des données sera couplée avec l'approche GoArrays (Rimour et al., 2005) pour proposer à l'utilisateur des couples de sondes afin d'augmenter le niveau de spécificité et de sensibilité des oligonucléotides.

Nous avons vu dans cette thèse que l'ingénierie dirigée par les modèles apporte un gain supplémentaire en termes de qualité et d'originalité des approches informatiques. Nous continuerons à appliquer les démarches innovantes pour l'intégration des systèmes d'informations destinés à la conception et à la gestion des données de biopuces à ADN. Nous avons montré qu'il est utile de centraliser l'information en utilisant une base de données commune à toutes nos applications bioinformatiques pour l'écologie microbienne. Elle permettra de factoriser le travail d'extraction et de traitement et de proposer une base de travail commune à toute la communauté de recherche plus généralement.

Le logiciel HiSpOD fera l'objet du développement d'une interface utilisateur. Une amélioration de l'affichage des résultats sera également réalisée. Une conception plus approfondie de sondes spécifiques à des protéines en utilisant la ferme locale de calcul et éventuellement la grille de calcul sera effectuée. Sachant que l'influence de certains critères de conception de sondes comme la température de fusion, le pourcentage GC ou la similarité BLAST n'est pas encore totalement maîtrisée, nous étudierons statistiquement l'effet de ces paramètres sur les oligonucléotides sélectionnés par notre logiciel lors d'expériences de validations biologiques.

Enfin, nos résultats ouvrent une voie prometteuse pour des recherches dans le développement de nouveaux algorithmes pour l'analyse des images de biopuces à ADN. Il s'agit en effet de proposer des outils innovants pour faire progresser le traitement des images de biopuces à grande échelle ou encore l'étude d'expression de plusieurs centaines de milliers de gènes comme par exemple en utilisant une classification par paire (Hanczar et al., 2007) et mieux comprendre l'interaction dans les réseaux de co-expression (Prifti et al., 2008). La combinaison des approches de biopuces phylogénétiques et de biopuces fonctionnelles pourrait contribuer à effectuer le lien entre la structure et la fonction, ce qui demeure l'un des enjeux majeurs de l'écologie microbienne. Bien évidemment, les progrès du séquençage ultra haut débit doivent être considérés pour gérer au mieux les masses de données à venir. Les approches de métagénomique sont également en plein essor et nécessitent de nouveaux développements bioinformatiques.

Nous aurons certainement besoin, de plus en plus, des architectures à hautes performances comme les grilles de calcul. Pour les approches locales, les GPUs (Graphics Processing Unit) peuvent également être une alternative prometteuse pour la génomique et la bioinformatique (Schatz et al., 2007 ; Sinnott-Armstrong et al., 2009 ; Rizk et Lavenier, 2009).

Références bibliographiques

A

Adebiyi, E.F. (2007) A comparative analysis of existing oligonucleotides selection algorithms for microarray technology, *African Journal of Biotechnology*, **6**, 1582-1586.

Afgan, E., Sathyanarayana, P. and Bangalore, P. (2006) Dynamic Task Distribution in the Grid for BLAST. *IEEE International Conference on Granular Computing*. 554-557.

Afzal, A., McGough, A.S. and Darlington, J. (2008) Capacity planning and scheduling in Grid computing environments, *Future Generation Computer Systems*, **24**, 404-414.

Alanen, M., Lilius, J., Porres, I. and Truscan, D. (2003) Realizing a Model Driven Engineering Process. TUCS, 27.

Alef, M., Fieseler, T., Freitag, S., Garcia, A., Grimm, C., Gürich, W., Mehammed, H., Schley, L., Schneider, O. and Volpato, G.L. (2009) Integration of multiple middlewares on a single computing resource, *Future Generation Computer Systems*, **25**, 268-274.

Alhir, S.S. (2003) Understanding the Model Driven Architecture (MDA), *Methods & Tools*, 16-24.

Altschul, S.F., Boguski, M.S., Gish, W. and Wootton, J.C. (1994) Issues in searching molecular sequence databases, *Nature Genetics*, **6**, 119-129.

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.

Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Research*, **25**, 3389-3402.

Amann, R.I., Krumholz, L. and Stahl, D.A. (1990) Fluorescent-oligonucleotide probing of whole cells for determinative, phylogenetic, and environmental studies in microbiology, *Journal of Bacteriology*, **172**, 762-770.

Amar, A., Bolze, R., Bouteiller, A., Chouhan, P.-K., Chis, A., Caniou, Y., Caron, E., Dail, H., Depardon, B., Desprez, F., Gay, J.-S., Le Mahec, G. and Su, A. (2006) DIET: New Developments and Recent Results. *CoreGRID Workshop on Grid Middleware (in conjunction with EuroPar2006)*. Dresden, Germany.

Ashelford, K.E., Chuzhanova, N.A., Fry, J.C., Jones, A.J. and Weightman, A.J. (2006) New Screening Software Shows that Most Recent Large 16S rRNA Gene Clone Libraries Contain Chimeras, *Applied Environmental Microbiology*, **72**, 5734–5741.

Ashelford, K.E., Weightman, A.J. and Fry, J.C. (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 Research*, **30**, 3481–3489.

Azmoodeh, M., Georgalas, N. and Fisher, S. (2005) Model-driven systems development and integration environment, *BT Technology Journal*, **23**, 96-110.

B

Baduel, L., Baude, F., Caromel, D., Contes, A., Huet, F., Morel, M. and Quilici, R. (2006) Programming, Composing, Deploying for the Grid. In Springer-Verlag (ed), *GRID COMPUTING: Software Environments and Tools*.

Bar-Or, C., Czosnek, H. and Koltai, H. (2007) Cross-species microarray hybridizations: a developing tool for studying species diversity, *Trends in Genetics*, **23**, 200-207.

Barton, G.J. (1998) Protein sequence alignment techniques., *Acta crystallographica. Section D, Biological crystallography*, **54**, 1139-1146.

Batory, D. (2006) Multilevel models in modeldriven engineering, product lines, and metaprogramming, *IBM Systems Journal*, **45**, 527-538.

Bayer, M.M. and Sinnott, R. (2005) Distributed BLAST in a Grid Computing Context. In Heidelberg, S.B. (ed), *Computational Life Sciences*. 241-252.

Bedell, J.A., Korf, I. and Gish, W. (2000) MaskerAid : a performance enhancement to RepeatMasker, *Bioinformatics*, **16**, 1040-1041.

Behr, T., Koob, C., Schedl, M., Mehlen, A., Meier, H., Knopp, D., Frahm, E., Obst, U., Schleifer, K., Niessner, R. and Ludwig, W. (2000) A nested array of rRNA targeted probes for the detection and identification of enterococci by reverse hybridization, *Systematic and applied microbiology*, **23**, 563-572.

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.

Bertis, V., Bolze, R., Desprez, F. and Reed, K. (2008) Large Scale Execution of a Bioinformatic Application on a Volunteer Grid. *Workshop on Parallel and Distributed Scientific and Engineering Computing (PDSEC)*. Miami, Florida.

Bézivin, J. (2004) In Search of a Basic Principle for Model Driven Engineering, *UPGRADE*, **V**, 21-24.

Bézivin, J. (2005) On the Unification Power of Models, *Software and Systems Modeling*, **4**, 171-188.

Bodrossy, L. and Sessitsch, A. (2004) Oligonucleotide microarrays in microbial diagnostics, *Current Opinion in Microbiology* **7**, 245-254.

Bodrossy, L., Stralis-Pavese, N., Konrad-Koszler, M., Weilharter, A., Reichenauer, T.G., Schofer, D. and Sessitsch, A. (2006) mRNA-based parallel detection of active methanotroph populations by use of a diagnostic microarray, *Applied and environmental microbiology*, **72**, 1672-1676.

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, *Environmental Microbiology*, **5**, 566-582.

Bokhari, S.H. and Sauer, J.R. (2006) Parallel algorithms for bioinformatics. In, *Parallel computing for bioinformatics and computational biology*. Wiley Interscience, 509-529.

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, *Applied and environmental microbiology*, **71**, 8042-8048.

Boukerche, A., Cristina, A. and Melo, M.A.d. (2006) Computational Molecular Biology. In Zomaya, A.Y. (ed), *Parallel Computing for Bioinformatics and Computational Biology*. John Wiley & Sons, Inc, 149.

Bozdech, Z., Zhu, J., Joachimiak, M.P., Cohen, F.E., Pulliam, B. and DeRisi, J.L. (2003) Expression profiling of the schizont and trophozoite stages of Plasmodium falciparum with a long-oligonucleotide microarray, *Genome Biology*, **4**, R9.

Bradley, R.K., Roberts, A., Smoot, M., Juvekar, S., Do, J., Dewey, C., Holmes, I. and Pachter, L. (2009) Fast Statistical Alignment, *PLoS Computational Biology* **5**, e1000392.

Bray, N. and Pachter, L. (2004) MAVID: constrained ancestral alignment of multiple sequences, *Genome Research*, **14**, 693-699.

Brodie, E.L., Desantis, T.Z., Joyner, D.C., Baek, S.M., Larsen, J.T., Andersen, G.L., Hazen, T.C., Richardson, P.M., Herman, D.J., Tokunaga, T.K., Wan, J.M. and Firestone, M.K. (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, *Proc Natl Acad Sci U S A*, **104**, 299-304.

C

Caramia, M. and Giordani, S. (2008) Resource allocation in grid computing: an economic model, *WSEAS Transactions on Computer Research*, **3**, 19-27.

Caron, E., Chis, A., Desprez, F. and Su, A. (2008) Design of plug-in schedulers for a GridRPC environment, *Future Generation Computer Systems.*, **24**, 46-57.

Caron, E., Desprez, F. and Le Mahec, G. (2008) Parallelization and Distribution Strategies of Large Bioinformatics Requests over the Grid. *International Conference on Algorithms and Architectures 2008 (ICA3PP 2008)*. Cyprus.

Carvalho, P.C., Glória, R.V., Miranda, A.B.d. and Degrave, W.M. (2005) Squid – a simple bioinformatics grid, *BMC Bioinformatics*, **6**.

Casey, R.M. (2005) BLAST Sequences Aid in Genomics and Proteomics. *Business Intelligence Network*.

Castiglioni, B., Rizzi, E., Frosini, A., Sivonen, K., Rajaniemi, P., Rantala, A., Mugnai, M.A., Ventura, S., Wilmotte, A., Boutte, C., Grubisic, S., Balthasart, P., Consolandi, C., Bordoni, R., Mezzelani, A., Battaglia, C. and De Bellis, G. (2004) Development of a universal microarray based on the ligation detection reaction and 16S rrna gene polymorphism to target diversity of cyanobacteria, *Appl Environ Microbiol*, **70**, 7161-7172.

Chappey, C., Danckaert, A., Dessen, P. and Hazout, S. (1991) MASH: an interactive program for multiple alignment and consensus sequence construction for biological sequences, *Bioinformatics*, **7**, 195-202.

Charbonnier, Y., Gettler, B., François, P., Bento, M., Renzoni, A., Vaudaux, P., Schlegel, W. and Schrenzel, J. (2005) A generic approach for the design of whole-genome oligoarrays, validated for genomotyping, deletion mapping and gene expression analysis on Staphylococcus aureus, *BMC Genomics*, **6**, 95.

Chaudhary, V., Liu, F., Matta, V. and Yang, L.T. (2005) Parallel Implementations of Local Sequence Alignment: Hardware and Software. In Sons, J.W. (ed), *Parallel Computing for Bioinformatics and Computational Biology*.

Chen, C. and Schmidt, B. (2005) An adaptive grid implementation of DNA sequence alignment, *Future Generation Computer Systems*, **21**, 988-1003.

Chen, H. and Sharp, B.M. (2002) Oliz, a suite of Perl scripts that assist in the design of microarrays using 50mer oligonucleotides from the 3' untranslated region, *BMC Bioinformatics*, **3**, 27.

Chen, Y.-P.P. (2005) *Bioinformatics Technologies*. Springer.

Chetty, M. and Buyya, R. (2002) Weaving Computational Grids: How Analogous Are they with Electrical Grids?, *Computing in Science and Engineering*, **4**, 61-71.

Chevassus-au-Louis, B. (2007) La biodiversité : un nouveau regard sur la diversité du vivant. Immensité et complexité, *Cahiers Agricultures* **16**.

Chou, H.-H., Hsia, A.-P., Mooney, D.L. and Schnable, P.S. (2004) PICKY: oligo microarray design for large genomes, *Bioinformatics*, **20**, 2893-2902.

Chung, W.-H., Rhee, S.-K., Wan, X.-F., Bae, J.-W., Quan, Z.-X. and Park, Y.-H. (2005) Design of long oligonucleotide probes for functional gene detection in a microbial community *Bioinformatics*, **21**, 4092-4100.

Cochrane, G., Akhtar, R., Aldebert, P., Althorpe, N., Baldwin, A., Bates, K., Bhattacharyya, S., Bonfield, J., Bower, L., Browne, P., Castro, M., Cox, T., Demiralp, F., Eberhardt, R., Faruque, N., Hoad, G., Jang, M., Kulikova, T., Labarga, A., Leinonen, R., Leonard, S., Lin, Q., Lopez, R., Lorenc, D., McWilliam, H., Mukherjee, G., Nardone, F., Plaister, S., Robinson, S., Sobhany, S., Vaughan, R., Wu, D., Zhu, W., Apweiler, R., Hubbard, T. and Birney, E. (2008) Priorities for nucleotide trace, sequence and annotation data capture at the Ensembl Trace Archive and the EMBL Nucleotide Sequence Database, *Nucleic Acids Research*, **36**, D5-D12.

Cody, E., Sharman, R., Rao, R.H. and Upadhyaya, S. (2008) Security in grid computing: A review and synthesis, *Decision Support Systems*, **44**, 749-764.

Cook, K.L. and Sayler, G.S. (2003) Environmental application of array technology: promise, problems and practicalities, *Current opinion in biotechnology*, **14**.

Curtis, T.P. and Sloan, W.T. (2004) Prokaryotic diversity and its limits: microbial community structure in nature and implications for microbial ecology, *Current Opinion in Microbiology*, **7**, 221-226.

D

Darling, A., Carey, L. and Feng, W. (2003) The Design, Implementation, and Evaluation of mpiBLAST. *4th International Conference on Linux Clusters: The HPC Revolution 2003*.

Datta, A. and Ebedes, J. (2005) Multiple Sequence Alignment in Parallel on a Cluster of Workstations. In Sons, J.W. (ed), *Parallel Computing for Bioinformatics and Computational Biology*.

DeLong, J.D.C. (1996) Defining Biodiversity, *Wildlife Society Bulletin*, **24**, 738-749.

Demchenko, Y. (2004) Virtual Organisations in Computer Grids and Identity Management, *Elsevier Information Security Technical Report*, **9**, 59-76.

Demchenko, Y., Mulmo, O., Gommans, L., Laat, C.d. and Wan, A. (2008) Dynamic security context management in Grid-based applications, *Future Generation Computer Systems*, **24**, 434-441.

DeSantis, T.Z., Brodie, E.L., Moberg, J.P., Zubieta, I.X., Piceno, Y.M. and Andersen, G.L. (2007) High-Density Universal 16S rRNA Microarray Analysis Reveals Broader Diversity than Typical Clone Library When Sampling the Environment, *Microbial Ecology*, **53**, 371-383.

DeSantis, T.Z., Stone, C.E., Murray, S.R., Moberg, J.P. and Andersen, G.L. (2005) Rapid quantification and taxonomic classification of environmental DNA from both prokaryotic and eukaryotic origins using a microarray, *FEMS Microbiol Lett*, **245**, 271-278.

Desprez, F. and Vernois, A. (2005) Simultaneous Scheduling of Replication and Computation for Bioinformatic Applications on the Grid. *CLADE 2005*.

Devos, D. and Valencia, A. (2001) Intrinsic errors in genome annotation, *Trends in Genetics*, **17**, 429-431.

Dharmadi, Y. and Gonzalez, R. (2004) DNA microarrays: experimental issues, data analysis, and application to bacterial systems, *Biotechnology progress*, **20**, 1309-1324.

Diedrich, K. (2002) DEODAS: A DEgenerate Oligonucleotide Design and Analysis System. *O'Reilly*.

Ding, Y., Chan, C.Y. and Lawrence, C.E. (2005) RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble, *RNA*, **11**, 1157–1166.

Do, C.B., Woods, D.A. and Batzoglou, S. (2006) CONTRAfold: RNA secondary structure prediction without physics-based models, *Bioinformatics*, **22**, e90–e98.

Dowd, S.E., Zaragoza, J., Rodriguez, J.R., Oliver, M.J. and Payton, P.R. (2005) Windows .NET Network Distributed Basic Local Alignment Search Toolkit (W.ND-BLAST), *BMC Bioinformatics*, **6**, 93.

Dufva, M. (2005) Fabrication of high quality microarrays, *Biomolecular Engineering*, **22**, 173-184.

E

Eddy, S.R. (2004) How do RNA folding algorithms work?, *Nature Biotechnology*, **22**, 1457-1458.

Egede, U., Harrison, K., Jones, R.W.L., Maier, A., Moscicki, J.T., Patrick, G.N., Soroko, A. and Tan, C.L. (2005) Ganga user interface for job definition and management. *Fourth International Workshop on Frontier Science: New Frontiers in Subnuclear Physics*. Milan, Italie, 367-374.

Ehrenreich, A. (2006) DNA microarray technology for the microbiologist: an overview, *Applied microbiology and biotechnology*, **73**, 255-273.

Emrich, S.J., Lowe, M. and Delcher, A.L. (2003) PROBEmer: a web-based software tool for selecting optimal DNA oligos, *Nucleic Acids Research*, **31**, 3746-3750.

Evertsz, E.M., Au-Young, J., Ruvolo, M.V., Lim, A.C. and Reynolds, M.A. (2001) Hybridization cross-reactivity within homologous gene families on glass cDNA microarrays, *Biothechniques*, **31**, 1182-1186.

F

Favre, J.-M., Estublier, J. and Blay-Fornarino, M. (2006) *L'ingénierie dirigée par les modèles*.

Favre, J.-M. and NGuyen, T. (2005) Towards a Megamodel to model software evolution through transformations, *Electronic Notes in Theoretical Computer Science*, **127**, 59-74.

Feng, S. and Tillier, E.R.M. (2007) A fast and flexible approach to oligonucleotide probe design for genomes and gene families, *Bioinformatics*, **23**, 1195-1202.

Ferguson, J.W. and Towns, J. (2001) The Alliance Grid, *Advances in Engineering Software*, **32**, 417-422.

Ferrari, B.C., Binnerup, S.J. and Gillings, M. (2005) Microcolony Cultivation on a Soil Substrate Membrane System Selects for Previously Uncultured Soil Bacteria *Applied and Environmental Microbiology*, **71**, 8714-8720.

Flynn, M. (1972) Some Computer Organizations and Their Effectiveness, *IEEE Transaction on Computer*, **C-21**, 948.

Flynn, M.J. (1966) Very high speed computing systems. *IEEE*. 1901-1909.

Flynn, M.J. and Rudd, K.W. (1996) Parallel architectures, *ACM Computing Surveys*, **28**, 67-70.

Fondement, F. and Silaghi, R. (2004) Defining Model Driven Engineering Processes. *Third International Workshop in Software Model Engineering (WiSME)*. Lisbon, Portugal.

Foster, I. (2006) Globus Toolkit Version 4: Software for Service-Oriented Systems, *Journal of Computer Science and Technology*, **21**, 513-520.

Foster, I. and Kesselman, C. (2004) *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.

Foster, I., Kesselman, C. and Tuecke, S. (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organization, *International Journal of High Performance Computing Applications*, **15**, 200-222.

Fotin, A.V., Drobyshev, A.L., Proudnikov, D.Y., Perov, A.N. and Mirzabekov, A.D. (1998) Parallel thermodynamic analysis of duplexes on oligodeoxyribonucleotide microchips, *Nucleic Acids Research*, **26**, 1515-1521.

Fujiwara, S. (2002) Extremophiles: Developments of their special functions and potential resources, *Journal of Bioscience and Bioengineering*, **94**, 518-525.

G

Galperin, M.Y. (2005) The Molecular Biology Database Collection: 2005 update, *Nucleic Acids Research*, **33**, D5-D24.

Galperin, M.Y. and Cochrane, G.R. (2009) Nucleic Acids Research annual Database Issue and the NAR online Molecular Biology Database Collection in 2009, *Nucleic Acids Research*, **37**, D1-D4.

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, *Environ Microbiol*, **10**, 836-850.

Gasieniec, L., Li, C.Y., Sant, P. and Wong, P.W.H. (2006) Efficient Probe Selection in Microarray Design. *CIBCB '06. 2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2006*. Toronto, Ont., 1-8.

Gentry, T.J., Wickham, G.S., Schadt, C.W., He, Z. and Zhou, J. (2006) Microarray Applications in Microbial Ecology Research, *Microbial ecology*, **52**, 159-175.

Gentzsch, W. (2002) Grid Computing: A New Technology for the Advanced Web, *Lecture notes in computer science*, **2326**, 1-15.

Gentzsch, W. (2008) Opinion - What clouds and grids can learn from each other, *International Science Grid This Week*.

Ghindilis, A.L., Smith, M.W., Schwarzkopf, K.R., Roth, K.M., Peyvan, K., Munro, S.B., Lodes, M.J., Stover, A.G., Bernards, K., Dill, K. and McShea, A. (2007) CombiMatrix oligonucleotide arrays: genotyping and gene expression assays employing electrochemical detection, *Biosensors & bioelectronics*, **22**, 1853-1860.

Gordon, P.M.K. and Sensen, C.W. (2004) Osprey: a comprehensive tool employing novel methods for the design of oligonucleotides for DNA sequencing and microarrays, *Nucleic Acids Research*, **32**, e133.

Gotelli, N.J. and Colwell, R.K. (2001) Quantifying biodiversity: procedures and pitfalls in the measurement and comparison of species richness, *Ecology Letters* **4**, 379-391.

Greenfield, J., Short, K., Cook, S., Kent, S. and Crupi, J. (2004) *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools (Paperback)*. Wiley.

Grimshaw, A., Humphrey, M., Knight, J.C., Nguyen-Tuong, A., Rowanhill, J., Wasson, G. and Basney, J. (2005) The Development of Dependable and Survivable Grids In Heidelberg, S.B. (ed), *Computational Science – ICCS 2005*. 729-737.

Guschin, D.Y., Mobarry, B.K., Proudnikov, D., Stahl, D.A., Rittmann, B.E. and Mirzabekov, A.D. (1997) Oligonucleotide microchips as genosensors for determinative and environmental studies in microbiology, *Applied and environmental microbiology*, **63**, 2397-2402.

H

Hamilton, A.J. (2005) Species diversity or biodiversity, *Journal of Environmental Management* **75**, 89-92.

Han, Y. and Youn, C.-H. (2009) A new grid resource management mechanism with resource-aware policy administrator for SLA-constrained applications, *Future Generation Computer Systems*, **25**, 768-778.

Hancock, J.M. and Armstrong, J.S. (1994) SIMPLE34: an improved and enhanced implementation for VAX and Sun computers of the SIMPLE algorithm for analysis of clustered repetitive motifs in nucleotide sequences, *Computer Applications in the Biosciences*, **10**, 67-70.

Hanczar, B., Zucker, J.-D., Henegar, C. and Saitta, L. (2007) Feature construction from synergic pairs to improve microarray-based classification, *Bioinformatics*, **23**, 2866-2872.

Hariri, S. and Parashar, M. (2004) *Tools and Environments for Parallel and Distributed Computing*.

Hawick, K.A. (1997) Trends in High Performance Computing. *Fourth IDEA Workshop, Magnetic Island*.

Hawksworth, D.L. (1995) *Biodiversity: Measurement and Estimation*. Springer; 1 edition (February 22, 2009).

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, *The ISME journal*, **1**, 67-77.

He, Z., Wu, L., Li, X., Fields, M.W. and Zhou, J. (2005) Empirical Establishment of Oligonucleotide Probe Design Criteria, *Applied and Environmental Microbiology*, **71**, 3753–3760.

Heitz, C., Thiemann, P. and Wölfle, T. (2007) Integration of an Action Language Via UML Action Semantics. In Heidelberg, S.B. (ed), *Trends in Enterprise Application Architecture*. 172-186.

Henikoff, T.M.R.J.G. and Henikoff, S. (2003) CODEHOP (COnsensus-DEgenerate Hybrid Oligonucleotide Primer) PCR primer design, *Nucleic Acids Research*, **31**, 3763-3766.

Hirschberg, D.S. (1975) A linear space algorithm for computing maximal common subsequences, *Communications of the ACM*, **18**, 341-343.

Hobbie, J.E., Daley, R.J. and Jasper, S. (1977) Use of nuclepore filters for counting bacteria by fluorescence microscopy., *Applied Environment Microbiollogy*, **33**, 1225-1228.

Hornshøj, H., Stengaard, H., Panitz, F. and Bendixen, C. (2004) SEPON, a Selection and Evaluation Pipeline for OligoNucleotides based on ESTs with a non-target Tm algorithm for reducing cross-hybridization in microarray gene expression experiments, *Bioinformatics*, **20**, 428-429.

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.

Hu, G., Llinás, M., Li, J., Preiser, P.R. and Bozdech, Z. (2007) Selection of long oligonucleotides for gene expression microarrays using weighted rank-sum strategy, *BMC Bioinformatics*, **8**, 350-362.

Huang, J.C., Morris, Q.D., Hughes, T.R. and Frey, B.J. (2005) GenXHC: a probabilistic generative model for cross-hybridization compensation in high-density genome-wide microarray data, *Bioinformatics*, **21**, i222-i231.

Hunter, P. (2003) Grid computing, *Network Security*, **4**, 15-16.

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.

J

Jacq, N., Blanchet, C., Combet, C., Cornillot, E., Duret, L., Kurata, K., Nakamura, H., Silvestre, T. and Breton, V. (2004) Grid as a bioinformatic tool, *Parallel Computing*, **30**, 1093–1107.

Jones, C.E., Brown, A.L. and Baumann, U. (2007) Estimating the annotation error rate of curated GO database sequence annotations, *BMC Bioinformatics*, **8**, 170.

Joseph, S.J., Hugenholtz, P., Sangwan, P., Osborne, C.A. and Janssen, P.H. (2003) Laboratory Cultivation of Widespread and Previously Uncultured Soil Bacteria, *Applied and Environmental Microbiology*, **69**, 7210-7215.

Jouault, F. and Kurtev, I. (2007) On the interoperability of model-to-model transformation languages, *Science of Computer Programming*, **68**, 114-137.

K

Kachalo, S., Arbieva, Z. and Liang, J. (2004) Assessing the Potential Effect of Cross-Hybridization on Oligonucleotide Microarrays In US, S. (ed), *Methods of Microarray Data Analysis III*. 185-198.

Kacsuk, P., Fahringer, T. and Nemeth, Z. (2007) *Distributed and Parallel Systems: From Cluster to Grid Computing*. Springer.

Kaderali, L. and Schliep, A. (2002) Selecting signature oligonucleotides to identify organisms using DNA arrays *Bioinformatics*, **18**, 1340-1349.

Kane, M.D., Jatkoe, T.A., Stumpf, C.R., Lu, J., Thomas, J.D. and Madore, S.J. (2000) Assessment of the sensitivity and specificity of oligonucleotide (50mer) microarrays, *Nucleic Acids Research*, **28**, 4552-4557.

Kawasaki, E.S. (2006) The End of the Microarray Tower of Babel: Will Universal Standards Lead the Way?, *Journal of Biomolecular Techniques*, **17**, 200–206.

Kelly, J.J., Siripong, S., McCormack, J., Janus, L.R., 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.

Kibbe, W.A. (2007) OligoCalc: an online oligonucleotide properties calculator, *Nucleic Acids Research*, **35**, W43-W46.

Kirk, J.L., Beaudette, L.A., Hart, M., Moutoglis, P., Klironomos, J.N., Lee, H. and Trevors, J.T. (2004) Methods of studying soil microbial diversity, *Journal of Microbiological Methods*, **58**, 169-188.

Klebanov, L., Chen, L. and Yakovlev, A. (2007) Revisiting adverse effects of cross-hybridization in Affymetrix gene expression data: do they matter for correlation analysis?, *Biology Direct*, 28.

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 Research*, **31**, 3423-3428.

Koblitz, B., Santos, N. and Pose, V. (2008) The AMGA Metagata Service, *Journal of Grid Computing*, **6**, 61-76.

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.

Kordon, F., Hugues, J. and Renault, X. (2008) From Model Driven Engineering to Verification Driven Engineering, *Lecture Notes in Computer Science*, 381–393.

Korf, I. and Gish, W. (2000) MPBLAST : improved BLAST performance with multiplexed queries, *Bioinformatics*, **16**, 1052-1053.

Kostic, T., Weilharter, A., Sessitsch, A. and Bodrossy, L. (2005) High-sensitivity, polymerase chain reaction-free detection of microorganisms and their functional genes using 70-mer oligonucleotide diagnostic microarray, *Analytical biochemistry*, **346**, 333-335.

Kranz, T. (2008) Symmetric Multiprocessing, and How Sun Makes it Scale.

Kreil, D.P., Russell, R.R. and Russell, S. (2006) Microarray oligonucleotide probes, *Methods in enzymology*, **410**, 73-98.

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, K.-i., 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.

Kumar, S.M. and Joshi, R.R. (2004) GBTK: a toolkit for grid implementation of BLAST. *High Performance Computing and Grid in Asia Pacific Region*. 378-382.

Kupczyk, M., Lichwała, R., Meyer, N., Palak, B., Płóciennik, M. and Wolniewicz, P. (2005) "Applications on demand" as the exploitation of the Migrating Desktop, **21**, 37-44.

Kyselková, M., Kopecký, J., Felföldi, T., Čermák, L., Omelka, M., Grundmann, G.L., Moënne-Loccoz, Y. and Ságová-Marečková, M. (2008) Development of a 16S rRNA gene-based prototype microarray for the detection of selected actinomycetes genera *Antonie van Leeuwenhoek*, **94**, 439-453.

L

Lassmann, T. and Sonnhammer, E.L.L. (2006) Kalign, Kalignvu and Mumsa: web servers for multiple sequence alignment, *Nucleic Acids Research*, **34**, W596-W599.

Laszewski, G.v. (2005) The Grid-Idea and Its Evolution, *Information Technologyu*, **47**, 319-329.

Laszewski, G.v., Foster, I., Gawor, J. and Lane, P. (2000) A Java Commodity Grid Kit, *Concurrency and Computation: Practice and Experience*, **13**, 645-662.

Launay, P. and Pazat, J.-L. (1997) A Framework for Parallel Programming in Java. INRIA, 13pp.

Laure, E., Fisher, S.M., Frohner, A., Grandi, C., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., Barroso, M., Buncic, P., Hemmer, F., Meglio, A.D. and Edlund, A. (2006) Programming the Grid with gLite, *Computational Methods in Science and Technology*, **12**, 33-45.

Lee, C., Grasso, C. and Sharlow, M.F. (2002) Multiple sequence alignment using partial order graphs, *Bioinformatics*, **18**, 452-464.

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 Microbiol Lett*, **246**, 133-142.

Leiske, D.L., Karimpour-Fard, A., Hume, P.S., Fairbanks, B.D. and Gill, R.T. (2006) A comparison of alternative 60-mer probe designs in an in-situ synthesized oligonucleotide microarray, *BMC Genomics*, **7**, 72.

Lemoine, S., Combes, F. and Crom, S.L. (2009) An evaluation of custom microarray applications: the oligonucleotide design challenge, *Nucleic Acids Research*, **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.

Li, F. and Stormo, G.D. (2001) Selection of optimal DNA oligos for gene expression arrays *Bioinformatics*, **17**, 1067-1076.

Li, K.-B. (2003) ClustalW-MPI: ClustalW analysis using distributed and parallel computing, *Bioinformatics*, **19**, 1585-1586.

Li, W. and Ying, W. (2006) Mprobe 2.0: Computer-Aided Probe Design for Oligonucleotide Microarray, *Applied Bioinformatics*, **5**, 181-186.

Li, X., He, Z. and Zhou, J. (2005) Selection of optimal oligonucleotide probes for microarrays using multiple criteria, global alignment and parameter estimation, *Nucleic Acids Research*, **33**, 6114-6123.

Lian, C.C., TANG, F., ISSAC, P. and KRISHNAN, A. (2005) GEL: Grid execution language, *Journal of Parallel and Distributed Computing*, **65**, 857-869.

López-Campos, G., Coiras, M., Sánchez-Merino, J.P., López-Huertas, M.R., Spiteri, I., Martín-Sánchez, F. and Pérez-Breña, P. (2007) Oligonucleotide microarray design for detection and serotyping of human respiratory adenoviruses by using a virtual amplicon retrieval software, *Journal of Virological Methods*, **145**, 127-136.

Loy, A., Arnold, R., Tischler, P., Rattei, T., Wagner, M. and Horn, M. (2008) probeCheck – a central resource for evaluating oligonucleotide probe coverage and specificity, *Environmental Microbiology*, **10**, 2894–2898.

Loy, A. and Bodrossy, L. (2006) Highly parallel microbial diagnostics using oligonucleotide microarrays *Clinica Chimica Acta*, **363**, 106-119.

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, *Appl Environ Microbiol*, **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", *Appl Environ Microbiol*, **71**, 1373-1386.

Luckow, A. and Schnor, B. (2008) Migol: A fault-tolerant service framework for MPI applications in the grid, *Future Generation Computer Systems*, **24**, 142-152.

Ludwig, W., Strunk, O., Westram, R., Richter, L., Meier, H., Yadhukumar, Buchner, A., Lai, T., Förster, S.S.G.J.W., Brettske, I., Gerber, S., Ginhart, A.W., Gross, O., Grumann, S., Hermann, S., Jost, R., König, A., Liss, T., Lüßmann, R., May, M., Nonhoff, B., Reichel, B., Strehlow, R., Stamatakis, A., Stuckmann, N., Vilbig, A., Lenke, M., Ludwig, T., Bode, A. and Schleifer, K.-H. (2004) ARB: a software environment for sequence data, *Nucleic Acids Research*, **32**, 1363-1371.

Lyngso, R.B., Zuker, M. and Pedersen, C.N.S. (1999) Fast evaluation of internal loops in RNA secondary structure prediction, *Bioinformatics*, **15**, 440-445.

M

Madigan, M.T., Martinko, J.M. and Parker, J. (1997) *Biology of microorganisms*. Prentice Hall; 9th edition (January 15, 2000).

Maglogiannis, I., Chatzioannou, A., Soldatos, J., Mylonakis, V. and Kanaris, Y. (2007) An Application Platform Enabling High Performance Grid Processing of Microarray Experiments. *the Twentieth IEEE International Symposium on Computer-Based Medical Systems*. IEEE Computer Society, 477-482.

Marcy, Y., Cousin, P.-Y., Rattier, M., Cerovic, G., Escalier, G., Béna, G., Guéron, M., McDonagh, L., Boulaire, F.l., Bénisty, H., Weisbuch, C. and Avarre, J.-C. (2008) Innovative integrated system for real-time measurement of hybridization and melting on standard format microarrays, *Biotechniques*, **44**, 913-920.

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, D.H. (2006) Revolutions in RNA Secondary Structure Prediction, *Journal of Molecular Biology*, **359**, 526-532

Mens, T. and Gorp, P.V. (2006) A Taxonomy of Model Transformation, *Electronic Notes in Theoretical Computer Science*, **152**, 125-142.

Mens, T., Gorp, P.V., Varró, D. and Karsai, G. (2006) Applying a Model Transformation Taxonomy to Graph Transformation Technology, *Electronic Notes in Theoretical Computer Science*, **152**, 143-159.

Militon, C., Rimour, S., Missaoui, M., Biderre, C., Barra, V., Hill, D., Mone, A., Gagne, G., Meier, H., Peyretaillade, E. and Peyret, P. (2007) PhylArray: phylogenetic probe design algorithm for microarray, *Bioinformatics*, **23**, 2550-2557.

Miller, S.R., Augustine, S., Olson, T.L., Blankenship, R.E., Selker, J. and Wood, A.M. (2005) Discovery of a free-living chlorophyll d-producing cyanobacterium with a hybrid proteobacterial/cyanobacterial small-subunit rRNA gene, *Proc Natl Acad Sci USA*, **102**, 850-855.

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. (2006) Optimisation des performances du programme mpiBLAST pour la parallélisation sur grille de calcul. LIMOS, Clermont-Fd, 9 pp.

Missaoui, M., Hill, D.R.C., Militon, C. and Peyret, P. (2007) Complete Backtranslation of Oligopeptides for Metabolic Pathways Exploration of Complex Environments using Functional Microarrays. *The 7th IEEE International Conference onBioinformatics and Bioengineering, 2007. BIBE 2007*. Harvard Medical School, Boston MA, USA, 1275-1279.

Missaoui, M., Hill, D.R.C. and Peyret, P. (2008) A comparison of algorithms for a complete backtranslation of oligopeptides, *International Journal of Computational Biology and Drug Design*, **1**, 26-38.

Missaoui, M., Hill, D.R.C. and Peyret, P. (2009) BLASTOG: a Parallel BLAST-based Software for Functional Microarrays Probe Design using the EGEE Computing Grid. *The 2009 International Conference on Grid Computing and Applications (GCA'09)*. Las Vegas, Nevada, USA, in press.

Moisander, P.H., Shiue, L., Steward, G.F., Jenkins, B.D., Bebout, B.M. and Zehr, J.P. (2006) Application of a nifH oligonucleotide microarray for profiling diversity of N2-fixing microorganisms in marine microbial mats, *Environmental Microbiology*, **8**, 1721-1735.

Moreira, A. (2004) Genetic algorithms for the imitation of genomic styles in protein backtranslation, *Theoretical Computer Science*, **322**, 297-312

Moreira, A. and Maass, A. (2004) TIP: protein backtranslation aided by genetic algorithms, *Bioinformatics*, **20**, 2148-2149

Moretti, S., Reinier, F., Poirot, O., Armougom, F., Audic, S., Keduas, V. and Notredame, C. (2006) PROTOGENE: turning amino acid alignments into bona fide CDS nucleotide alignments, *Nucleic Acids Research*, **34**, W600-W603.

Morgulis, A., Gertz, E.M., Schaffer, A.A. and Agarwala, R. (2006) A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences, *Journal of Computational Biology*, **13**, 1028-1040.

Mrowka, R., Schuchhardt, J. and Gille, C. (2002) Oligodb—interactive design of oligo DNA for transcription profiling of human genes, *Bioinformatics*, **18**, 1686-1687.

N

Navarro, G. (2001) A guided tour to approximate string matching, *ACM Computing Surveys*, **33**, 31-88.

Navin, N., Grubor, V., Hicks, J., Leibu, E., Thomas, E., Troge, J., Riggs, M., Lundin, P., Månér, S., Sebat, J., Zetterberg, A. and Wigler, M. (2006) PROBER: oligonucleotide FISH probe design software, *Bioinformatics*, **22**, 2437-2438.

Needleman, S.B. and Wunsch, C. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, **48**, 443-453

Neufeld, J.D., Mohn, W.W. 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.

Nielsen, H.B. and Knudsen, S. (2002) Avoiding cross hybridization by choosing nonredundant targets on cDNA arrays *Bioinformatics*, **18**, 321-322.

Nielsen, H.B., Wernersson, R. and Knudsen, S. (2003) Design of oligonucleotides for microarrays and perspectives for design of multi-transcriptome arrays, *Nucleic Acids Research*, **31**, 3491-3496.

Nocker, A., Burr, M. and Camper, A.K. (2007) Genotypic Microbial Community Profiling: A Critical Technical Review *Microbial Ecology*, **54**, 276-289.

Nordberg, E.K. (2005) YODA: selecting signature oligonucleotides *Bioinformatics*, **21**, 1365-1370.

Notredame, C., Higgins, D.G. and Heringa, J. (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment, *Journal of Molecular Biology*, **302**, 205-217.

O

Oksanen, M. and Pietarinen, J. (2004) *Phylosophy and biodiversity*.

Oroguchi, T., Inoue, T., Tomabechi, I., Tago, Y. and Makino, T. (2005) Development of new method of primer design based on Genomewide for SNP typing under problem solving environment on grid. *First International Conference on e-Science and Grid Computing*. Japan, 6pp.

P

Page, A.J. and McInerney, J.O. (2005) A High-Throughput Bioinformatics Distributed Computing Platform. *IEEE Symposium on Computer-Based Medical Systems*. IEEE Computer Society, 377-382

Palmer, C., Bik, E.M., Eisen, M.B., Eckburg, P.B., Sana, T.R., Wolber, P.K., Relman, D.A. and Brown, P.O. (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.

Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence comparison, *Procedings of the National Academy of Science*, **85**, 2444-2448.

Pei, J., Sadreyev, R. and Grishin, N.V. (2003) PCMA: fast and accurate multiple sequence alignment based on profile consistency *Bioinformatics*, **19**, 427-428.

Peplies, J., Lachmund, C., Glöckner, F.O. and Manz, W. (2006) A DNA Microarray Platform Based on Direct Detection of rRNA for Characterization of Freshwater Sediment-Related Prokaryotic Communities, *Applied and Environmental Microbiology*, **72**, 4829-4838.

Pesole, G., Attimonelli, M. and Liuni, S. (1988) A backtranslation method based on codon usage strategy, *Nucleic Acids Research*, **16**, 1715-1728.

Petzold, E., Merkle, D., Middendorf, M., Haeseler, A.v. and Schmidt, H.A. (2006) Phylogenetic Parameter Estimation on COWs. In Zomaya, A.Y. (ed), *Parallel Computing for Bioinformatics and Computational Biology*. Wiley Interscience, 347-368.

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.

Pozhitkov, A.E., Tautz, D. and Noble, P.A. (2007) Oligonucleotide microarrays: widely applied—poorly understood, *Briefings in Functional Genomics and Proteomics*, **6**, 141-148.

Prifti, E., Zucker, J.-D., Clement, K. and Henegar, C. (2008) FunNet: an integrative tool for exploring transcriptional interactions, *Bioinformatics*, **24**, 2636-2638.

Priscu, J.C., Fritsen, C.H., Adams, E.E., Giovannoni, S.J., Paerl, H.W., McKay, C.P., Doran, P.T., Gordon, D.A., Lanoil, B.D. and Pinckney, J.L. (1998) Perennial Antarctic lake ice: an oasis for life in a polar desert, *Science*, **280**, 2095-2098

R

Rahmann, S. (2003) Fast large scale oligonucleotide selection using the longest common factor approach, *Journal of Bioinformatics and Computational Biology*, **1**, 343-361.

Raih, M.F., Sharum, M.Y., Moktar, R.M.R., Isa, M.N.M., Kian, N.L., Mahadi, N.M. and Mohamed, R. (2005) EMASGRID: An NBBnet Grid Initiative for a Bioinformatics and Computational Biology Services Infrastructure in Malaysia In Heidelberg, S.B. (ed), *Grid Computing in Life Science*. 117-124.

Relógio, A., Schwager, C., Richter, A., Ansorge, W. and Valcárcel, J. (2002) Optimization of oligonucleotide-based DNA microarrays *Nucleic Acids Research*, **30**, e51.

Reymond, N., Duret, H.C.L., Calevro, F., Beslon, G. and Fayard, J.-M. (2004) ROSO: optimizing oligonucleotide probes for microarrays, *Bioinformatics*, **20**, 271-273.

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.

Rizk, G. and Lavenier, D. (2009) GPU Accelerated RNA Folding Algorithm. In Heidelberg, S.B. (ed), *Computational Science – ICCS 2009*. 1004-1013.

Rose, T.M., Schultz, E.R., Henikoff, J.G., Pietrokovski, S., McCallum, C.M. and Henikoff, S. (1998) Consensus-degenerate hybrid oligonucleotide primers for amplification of distantly related sequences, *Nucleic Acids Research*, **26**, 1628–1635.

Rougemaille, S., Migeon, F., Millan, T. and Gleizes, M.-P. (2009) Methodology Fragments Definition in SPEM for Designing Adaptive Methodology: A First Step, *Lecture Notes in Computer Science*, 74-85.

Rouillard, J.-M., Zuker, M. and Gulari, E. (2003) OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach, *Nucleic Acids Research*, **31**, 3057-3062.

S

Sambrook, J. and Russell, D.W. (2001) Empirical Measurement of Melting Temperature. In, *Molecular Cloning*. Cold Spring Harbor Laboratory Press.

Sanguin, H., Kroneisen, L., Gazengel, K., Kyselková, M., Remenant, B., Prigent-Combaret, C., Grundmann, G.L., 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., Remenant, B., Dechesne, A., Thioulouse, J., Vogel, T.M., Nesme, X., Moënne-Loccoz, Y. and Grundmann, G.L. (2006) Potential of a 16S rRNA-Based Taxonomic Microarray for Analyzing the Rhizosphere Effects of Maize on Agrobacterium spp. and Bacterial Communities, *Applied and Environmental Microbiology*, **72**, 4302–4312.

SantaLucia, J., Allawi, H.T. and Seneviratne, P.A. (1996) Improved Nearest-Neighbor Parameters for Predicting DNA Duplex Stability, *Biochemistry*, **35**, 3555-3562.

Santos, N. and Koblitza, B. (2006) Metadata services on the Grid *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, **559**, 53-56

Sarkar, S. (2005) *Biodiversity and Environmental Philosophy: An Introduction*.

Schatz, M.C., Trapnell, C., Delcher, A.L. and Varshney, A. (2007) High-throughput sequence alignment using Graphics Processing Units, *BMC Bioinformatics*, **8**, 474.

Schena, M., Shalon, D., Davis, R.W. and Brown, P.O. (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray, *Science*, **270**, 467-470.

Schretter, C. and Milinkovitch, M.C. (2005) OLIGOFAKTORY: a visual tool for interactive oligonucleotide design *Bioinformatics*, **22**, 115-116.

Schwartz, S., Kent, W.J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R.C., Haussler, D. and Miller, W. (2003) Human–Mouse Alignments with BLASTZ, *Genome Research*, **13**, 103-107.

Sellers, P.H. (1974) On the theory and computation of evolutionary distances, *Journal of Applied Mathematics*, **26**, 787-793.

Sendall, S. and Kozaczynski, W. (2003) Model Transformation - the Heart and Soul of Model-Driven Software Development., *IEEE Software*, **20**, 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, *New Phytol*, **171**, 719-735.

Shiers, J. (2009) Gridnext term today, clouds on the horizon, *Computer Physics Communications*, **180**, 559-563.

Simmler, H., Singpiel, H. and Männer, R. (2003) Real-time primer design for DNA chips. *17th International Symposium on Parallel and Distributed Processing*. IEEE Computer Society, 153.

Sinnott-Armstrong, N.A., Greene, C.S., Cancare, F. and Moore, J.H. (2009) Accelerating epistasis analysis in human genetics with consumer graphics hardware, *BMC Research Notes*, **2**, 149.

Small, J., Call, D.R., Brockman, F.J., Straub, T.M. and Chandler, D.P. (2001) Direct detection of 16S rRNA in soil extracts by using oligonucleotide microarrays, *Appl Environ Microbiol*, **67**, 4708-4716.

Smith, J.J., Tow, L.A., Stafford, W., Cary, C. and Cowan, D.A. (2006) Bacterial Diversity in Three Different Antarctic Cold Desert Mineral Soils, *Microbial Ecology*, **51**, 413–421.

Smith, M., Schmidt, M., Fallenbeck, N., Dörnemann, T., Schridde, C. and Freisleben, B. (2009) Secure on-demand grid computing, *Future Generation Computer Systems*, **25**, 315-325.

Smith, T.F. and Waterman, M.S. (1981) Identification of common molecular subsequences *Journal of Molecular Biology*, **147**, 195-197.

Staley, J.T. and Konopka, A. (1985) Measurement of in Situ Activities of Nonphotosynthetic Microorganisms in Aquatic and Terrestrial Habitats, *Annual Review of Microbiology*, **39**, 321-346.

Stekel, D. (2003) *Microarray Bioinformatics*. Cambridge University Press.

Stenberg, J., Nilsson, M. and Landegren, U. (2005) ProbeMaker: an extensible framework for design of sets of oligonucleotide probes, *BMC Bioinformatics*, **6**, 229.

Stoffels, M., Castellanos, T. and Hartmann, A. (2001) Design and Application of New 16S rRNA-targeted Oligonucleotide Probes for the Azospirillum-Skermanella-Rhodocista-Cluster, *Systematic and Applied Microbiology*, **24**, 83-97.

Stratowa, C. (2003) Novel Framework for Distributed Storage and Analysis of Microarray Data in the Terabyte Range: An Alternative to BioConductor. *DSC 2003*.

Stüber, K. (1986) Nucleic acid secondary structure prediction and display, *Nucleic Acids Research*, **14**, 317-326.

Sun, Y., Zhao, S., Yu, H., Gao, G. and Luo, J. (2007) ABCGrid: Application for Bioinformatics Computing Grid, *Bioinformatics*, **23**, 1175-1177.

Suyama, M., Torrents, D. and Bork, P. (2006) PAL2NAL: robust conversion of protein sequence alignments into the corresponding codon alignments, *Nucleic Acids Research*, **34**, W609-W612.

T

Talbi, E.-G. and Zomaya, A.Y. (2008) *Grid Computing for Bioinformatics and Computational Biology*.

Taniguchia, M., Miuraa, K., Iwaoa, H. and Yamanakab, S. (2001) Quantitative Assessment of DNA Microarrays—Comparison with Northern Blot Analyses *Genomics*, **71**, 34-39.

Tardieu, H., Rochfeld, A. and Coletti, R. (1983) *La Méthode MERISE Tome 1 - Principes et outils*. Editions d'Organisation.

Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CLUSTALW: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research*, **22**, 4673–4680.

Timlin, J.A. (2006) Scanning microarrays: current methods and future directions, *Methods Enzymol*, **411**, 79-98.

Tolstrup, N., Nielsen, P.S., Kolberg, J.G., Frankel, A.M., Vissing, H. and Kauppinen, S. (2003) OligoDesign: optimal design of LNA (locked nucleic acid) oligonucleotide capture probes for gene expression profiling, *Nucleic Acids Research*, **31**, 3758-3762.

Tomiuk, S. and Hofmann, K. (2001) Microarray probe selection strategies *Briefings in Bioinformatics*, **2**, 329-340.

Torsvik, V. and Øvreås, L. (2002) Microbial diversity and function in soil: from genes to ecosystems *Current Opinion in Microbiology*, **5**, 240-245.

Trelles, O. (2000) On the parallelisation of bioinformatics applications, *Brief in Bioinformatics*, **2**, 181-194.

Trombetti, G.A., 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.

Turner, P., Mamo, G. and Karlsson, E.N. (2007) Potential and utilization of thermophiles and thermostable enzymes in biorefining, *Microbial Cell Factories*, **6**, 9-32.

V

Venter, J.C., Remington, K., Heidelberg, J.F., Halpern, A.L., Rusch, D., Eisen, J.A., Wu, D., Paulsen, I., Nelson, K.E., Nelson, W., Fouts, D.E., Levy, S., Knap, A.H., Lomas, M.W., Nealson, K., White, O., Peterson, J., Hoffman, J., Parsons, R., Baden-Tillson, H., Pfannkoch, C., Rogers, Y.-H. and Smith, H.O. (2004) Environmental Genome Shotgun Sequencing of the Sargasso Sea, *Science*, **304**, 66-74.

Vieites, J.M., Guazzaroni, M.-E., Beloqui, A., Golyshin, P.N. and Ferrer, M. (2008) Metagenomics approaches in systems microbiology, *FEMS microbiology reviews*, **33**, 236-255.

W

Wackett, L.P. and Hershberger, D.C. (2000) *Biocatalysis and biodegradation.Microbial transformation of organic compounds*.

Wagner, M., Nielsen, P.H., Loy, A., Nielsen, J.L. and Daims, H. (2006) Linking microbial community structure with function: fluorescence in situ hybridization-microautoradiography and isotope arrays, *Current Opinion in Biotechnology*, **17**, 83-91.

Wagner, M., Smidt, H., Loy, A. and Zhou, J. (2007) Unravelling Microbial Communities with DNA-Microarrays: Challenges and Future Directions, *Microbial Ecology*, **53**, 498-506.

Wang, X. and Seed, B. (2003) Selection of Oligonucleotide Probes for Protein Coding Sequences, *Bioinformatics*, **19**, 796-802.

Wernersson, R. and Nielsen, H.B. (2005) OligoWiz 2.0—integrating sequence feature annotation into the design of microarray probes, *Nucleic Acids Research*, **33**, W611-W615.

Wernersson, R. and Pedersen, A.G. (2003) RevTrans: multiple alignment of coding DNA from aligned amino acid sequences, *Nucleic Acids Research*, **31**, 3537-3539.

Wetmur, J.G. (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, W.B., Coleman, D.C. and Wiebe, W.J. (1998) Prokaryotes: the unseen majority, *Proc Natl Acad Sci USA*, 6578-6583.

Wiese, K.C., Deschenes, A.A. and A.G Hendriks (2008) RnaPredict—An Evolutionary Algorithm for RNA Secondary Structure Prediction, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **5**, 25-41.

Wiese, K.C. and Hendriks, A. (2006) Comparison of P-RnaPredict and mfold—algorithms for RNA secondary structure prediction *Bioinformatics*, **22**, 934-942.

Willard, B. (2007) UML for systems Engineering, *Computer Standards and Interfaces*, **29**, 69-81.

Willey, J., Sherwood, L. and Woolverton, C. (2007) *Microbiology*. McGraw-Hill Science/Engineering/Math; 7 edition (January 12, 2007).

Wilson, K.H., Wilson, W.J., Radosevich, J.L., DeSantis, T.Z., Viswanathan, V.S., Kuczmarski, T.A. and Andersen, G.L. (2002) High-density microarray of small-subunit ribosomal DNA probes, *Appl Environ Microbiol*, **68**, 2535-2541.

Wolniewicz, P., Meyer, N., Stroiński, M., Stuempert, M., Kornmayer, H., Polak, M. and Gjermundrød, H. (2007) Accessing Grid Computing Resources with g-Eclipse Platform, *Computational Methods in Science and Technologie*, **13**, 131-141.

Wootton, J.C. and Federhen, S. (1993) Statistics of local complexity in amino acid sequences and sequence databases, *Computers Chemistery*, **17**, 149-163.

Wootton, J.C. and Federhen, S. (1996) Analysis of compositionally biased regions in sequence databases, *Methods in Enzymology*, **266**, 554-571.

Wren, J.D., Kulkarni, A., Joslin, J. and Butow, R.A.H., R.G. (2002) Cross-hybridization on PCR-spotted microarrays, *IEEE Engineering in Medicine and Biology Magazine*, **21**, 71-75.

Wu, C., Carta, R. and Zhang, L. (2005) Sequence dependence of cross-hybridization on short oligo microarrays, *Nucleic Acids Research*, **33**, e84.

Wu, L., Thompson, D.K., Liu, X., Fields, M.W., Bagwell, C.E., Tiedje, J.M. and Zhou, J. (2004) Development and evaluation of microarray-based whole-genome hybridization for detection of microorganisms within the context of environmental applications, *Environ Sci Technol*, **38**, 6775-6782.

Wu, X. and Tseng, C.-W. (2005) Searching Sequence Databases Using High Performance BLASTs In Sons, J.W. (ed), *Parallel Computing for Bioinformatics and Computational Biology*

X

Xu, D., Xu, Y., Li, G. and Zhou, J. (2000) A Computer Program for Generating Gene-Specific Fragments for Microarrays, *Currents in Computational Molecular Biology*, 3-4.

Y

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.

Z

Zabatta, F. and Ying, K. (1998) A Thread Performance Comparison: Windows NT and Solaris on A Symmetric Multiprocessor. *2nd USENIX Windows NT Symposium*. 57–66.

Zhang, L., Coombes, K.R. and Xiao, L. (2004) Quantification of Cross Hybridization on Oligonucleotide Microarrays In US, S. (ed), *Methods of Microarray Data Analysis III*. 175-184.

Zhang, Y., Zhang, X., Liu, X., Xiao, Y., Qu, L., Wu, L. and Zhou, J. (2007) Microarray-based analysis of changes in diversity of microbial genes involved in organic carbon decomposition following land use/cover changes, *FEMS microbiology letters*, **266**, 144-151.

Zhang, Z., Schaffer, A.A., Miller, W., Madden, T.L., Lipman, D.J., Koonin, E.V. and Altschul, S.F. (1998) Protein sequence similarity searches using patterns as seeds, *Nucleic Acids Research*, **26**, 3986-3990.

Zhou, J. and Thompson, D.K. (2002) Challenges in applying microarrays to environmental studies *Current Opinion in Biotechnology*, **13**, 204-207.

Zhu, D., Fofanov, Y., Willson, R.C. and Fox, G.E. (2006) A parallel computing algorithm for 16S rRNA probe design, *Journal of Parallel and Distributed Computing*, **66**, 1546-1551.

Zomaya, A.Y. (2006) *Parallel Computing for Bioinformatics and Computational Biology*.

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 Research*, **31**, 3406-3415.

Liste des publications

Revues internationales (3)

J-F Brugère, A. Mihajlovski, **M. Missaoui**, P. Peyret. *Tools for stools: assessing intestinal microbiota using molecular diagnostics*. Expert Review of Molecular Diagnostics. **2009**. 9(4): 353-365.

M. Missaoui, D.R.C. Hill, P. Peyret. *A comparison of algorithms for a complete backtranslation of oligopeptides*. International Journal of Computational Biology and Drug Design. **2008** Vol. 1, No.1 pp. 26-38

C. Militon, S. Rimour**, M. Missaoui**, C. Biderre, V. Barra, D.R.C. Hill, A. Mone, G. Gagne, H. Meier, E. Peyretaillade, P. Peyret. *PhylArray: phylogenetic probe design algorithm for microarray*. Bioinformatics. **2007**. 23: 2550-2557.

Conférences internationales avec publication d'article et comité de lecture (2)

M. Missaoui, C. Militon, D.R.C. Hill, P. Peyret. *BLASTOG: a Parallel BLAST-based software for Functional Microarrays Probe Design using the EGEE Computing Grid*. 2009. GCA'09 - The 2009 International Conference on Grid Computing and Applications at Monte Carlo Resort, Las Vegas, Nevada, USA (**July 13-16, 2009**).). pp. 126-132.

M. Missaoui, C. Militon, D.R.C. Hill, P. Peyret. *Complete Backtranslation of Oligopeptides for Metabolic Pathways Exploration of Complex Environments Using Functional Microarrays*. The IEEE 7th International Conference on Bioinformatics and Biomedical Engineering at Harvard Medical School. (**October 14-17, 2007**). pp.1275–1279.

Conférences internationales avec comité de lecture et publication de résumé (1)

S.Terat, C. Militon, C. Biderre, S. Rimour, A. Moné, **M. Missaoui**, O. Gonçalves, E. Peyretaillade, P. Peyret. *New algorithm for development of oligonucleotide microarrays dedicated to study biodiversity of metabolic pathways in polluted soils*. 9th Symposium on Bageco 9. Bacterial Genetics and Ecology. Microbial Community Networks. (**June 23-27 2007**). 035: pp. 52

Conférences nationales avec comité de lecture et publication de résumé (3)

E. Dugat-Bony, **M. Missaoui**, O. Bouzid, E. Dumas, S. Terrat, C. Biderre-Petit, P. Peyret. *Développement de Biopuces ADN outils moléculaires diagnostics pour l'évaluation des potentialités métaboliques microbiennes d'écosystèmes contaminés : Projet EVASOL (ANR-PRECODD)*. 2èmes rencontres nationales de la Recherche sur les sites et sols pollués, Paris, **Octobre 2009**.

C. Militon, S. Terrat, C. Biderre, S. Rimour, A.C Lehours, A. Moné., **M. Missaoui**, O. Goncalves, E. Peyretaillade, G. Gagne , C. Maillet, D.Morgavi, E. Forano, G. Fonty, P. Peyret. *Etude des voies métaboliques des populations microbiennes d'environnements complexes par une approche biopuce ADN*. 7ème congrès national de la SFM. Nantes **30, 31 mai et 1er Juin 2007**.

C. Militon, S. Terrat, C. Biderre, S. Rimour, A. Moné., **M. Missaoui**, G. Gagne, M. Bonnet, O. Goncalves, E. Peyretaillade, P. Peyret. *Développement de nouveaux algorithmes pour la conception de biopuces ADN taxonomiques et fonctionnelles permettant d'appréhender la diversité microbienne d'environnements complexes*. Bioinformatique et biodiversité microbienne (Pasteur Génopole Ile de France, ReNaBI, SFM) **6 Mars 2007**.

Séminaires (5)

M. Missaoui, S. Bornes, O. Goncalves, S. Rimour, D.R.C. Hill, E. Peyretaillade, P. Peyret. *Système d'Information pour la plateforme « Conception et Analyse de Biopuces ».* Séminaire LifeGrid 2008. **24 Novembre 2008**.

M. Missaoui, D.Hill, P. Peyret. *Microarrays development using the Auvergrid platform*. Interdisciplinary Congress organized by the Federation of Research Environment "site Clermont-Ferrand" **23 October 2008**.

M. Missaoui, D.R.C. Hill, P. Peyret. *Développement d'algorithmes pour les grilles de calculs*. Séminaire AFEM : Biopuces et Ecologie Microbienne, Clermont-Ferrand les **19-20 Octobre 2006**

C. Biderre-Petit, V. Lasbat, X. Brottel, A.C. Lehours, S. Terrat, A. Moné, C. Militon, B. Chebance, **M. Missaoui,** O. Goncalves, E. Peyretaillade, D. Morgavi, G. Fonty, P. Peyret. *Etude de la diversité bactérienne et caractérisation de voies métaboliques du méthane par une approche biopuce à ADN*. Association Francophone d'Ecologie Microbienne (AFEM), congrès Biopuces et écologie microbienne, Clermont-Ferrand, **19-20 Octobre 2006**.
C. Militon, S.Terrat, **M. Missaoui,** A. Moné, C. Vachlard, A. Belkorchia, J. Brunellière, M. Bonnet, B. Chebance, G. Gagne, O. Goncalves, C. Petit, P. Veisseire, E. Peyretaillade, J. Troquet, P. Peyret. *Caractérisation de microorganismes et de voies métaboliques épuratrices de sols pollués aux hydrocarbures par des approches biopuces ADN*. Biodépollution et environnement : Savoir et savoir-faire, Paris les **12-13 Septembre 2006**.

Rapports de recherche (3)

M. Missaoui, C. Militon, D. Hill, Ch. Gouinaud and P. Peyret. *PRT: Parallel program for a full backtranslation of oligopeptides*. [LIMOS/RR-07-11]. http://www.isima.fr/limos/publi/RR-07-11.pdf

M. Missaoui, D.RC. Hill and P.Peyret. *DegenRev: Degeneracy-Based Full Backtranslation Algorithm for Oligopeptide*. [LIMOS/RR-07-10]. http://www.isima.fr/limos/publi/RR-07-10.pdf

M. Missaoui. *Optimisation des performances du programme mpiBLAST pour la parallélisation sur grille de calcul*. [LIMOS/RR-06-10]. http://www.isima.fr/limos/publi/RR-06-10.pdf

RESUME

Les microorganismes constituent la plus grande diversité du monde vivant et restent encore largement méconnus. La compréhension du fonctionnement des écosystèmes et les rôles joués par les microorganismes reste un enjeu majeur de l'écologie microbienne. Le développement de nouvelles approches de génomique permet d'appréhender la complexité de ces systèmes. Dans ce contexte, les puces à ADN représentent des outils à haut débit de choix capables de suivre l'expression ou la présence de plusieurs milliers de gènes en une seule expérience. Cependant, l'une des étapes les plus difficiles, de part sa complexité et la quantité de données à traiter, est la sélection de sondes qui doivent être à la fois sensibles et spécifiques. Pour répondre à ces besoins en termes de performance et de qualité, d'une part, nous avons développé un algorithme de conceptions de sondes pour biopuces phylogénétiques que nous avons entièrement déployé sur la grille de calcul européenne EGEE, et d'autre part, nous avons proposé un deuxième algorithme pour biopuces fonctionnelles que nous avons déployé sur un cluster. Les deux algorithmes ont nécessité une phase importante d'ingénierie logicielle. Nous avons donc proposé une démarche d'ingénierie dirigée par les modèles (IDM) et notamment de transformation de modèle pour résoudre le problème de traduction inverse d'oligopeptide. Les deux algorithmes sont destinés à une utilisation massive et permettent de concevoir tout type de sondes.

Mots-clés: Conception de sondes, Puces à ADN, Grille de calcul, IDM, Ecologie microbienne.

ABSTRACT

Microorganisms represent the largest diversity of the living beings and still largely unknown. Understanding ecosystems working mechanisms and the roles of microorganisms is a great challenge in microbial ecology. The development of new genomic approaches allows us to comprehend the complexity of those systems. In this context, DNA microarrays represent high-throughput tools able to show the expression or the presence of several thousands of genes on a single experiment. However, one of the hardest steps in microarray utilization is the probe selection because of its complexity and the huge amount of data to compute. Obtained probes should be specific and sensitive. To reach the goals of performance and quality, we have developed an algorithm for phylogenetic microarrays and we have deployed it on the European Computing Grid EGEE. We have also developed a new algorithm for functional microarrays and we have deployed it onto a cluster. The two algorithms needed an important step of software engineering. In this context, we have proposed a new method based on model driven engineering (MDE) and particularly on model transformation to solve the problem of backtranslation of oligopeptides. The two algorithms are intended to be used massively and allow bioinformaticians and biologists to design all kinds of probes.

Keywords: Probe design, Microarrays, Grid computing, MDE, Microbial ecology.