N° d'ordre 139-99 **Année 1999**

THÈSE

présentée

devant l'UNIVERSITÉ CLAUDE BERNARD – LYON 1

pour l'obtention

du DIPLÔME DE DOCTORAT

(arrêté du 30 mars 1992)

présentée et soutenue publiquement le

31 mai 1999

par

M. Christophe BLANCHET

TITRE :

Logiciel MPSA et ressources bioinformatiques client-serveur Web dédiés à l'analyse de séquences de protéine.

Directeur de thèse : Pr Gilbert DELEAGE

JURY : Dr Marc-André DELSUC, Rapporteur
Dr Philippe DESSEN, Rapporteur
Dr Jean-Jacques CODANI
Pr Alain-Jean COZZONE, Président
Pr Gilbert DELÉAGE

Avant-Propos

Avant de commencer, j'aimerais attirer l'attention du lecteur sur deux notions.

Tout d'abord je trouve qu'il est toujours très difficile de présenter un logiciel ou un site Web par écrit : que ce soit dans un document relié, avec des transparents ou avec des diapositives. C'est pourquoi, je ne saurais que conseiller au lecteur d'essayer les logiciels et de visiter les sites Web que je présente dans cet ouvrage. Notamment en ce qui concerne MPSA, le lecteur pourra trouver toutes les informations disponibles sur le site Web (http://www.ibcp.fr/mpsa) et y télécharger un exécutable pour son système d'exploitation.

La deuxième notion sur laquelle je voudrais attirer votre attention est la mise sur le Web de cette thèse. En effet, cet ouvrage traitant de bioinformatique et du World-Wide Web, il contient de nombreuses références à d'autres documents répartis sur Internet. C'est pourquoi je publierai la version finale de ma thèse au format HTML sur ma page personnelle (http://www.ibcp.fr/~blanchet).

Remerciements

En premier lieu mes remerciements vont à mes rapporteurs, les Dr Phillipe Dessen et Marc-André Delsuc, pour avoir accepté de juger le travail que j'ai effectué au cours de ma thèse. Mes remerciements sont également adressés au Dr Jean-Jacques Codani, au Pr Alain-Jean Cozzone et au Pr Gilbert Deléage pour leur participation à mon jury.

J'ai une attention particulière pour Gilbert Deléage qui m'a accueilli depuis mon stage de DEA dans son équipe, le Laboratoire de Conformation des Protéines (sans oublier François Penin). Sans lui, je ne serais peut-être pas revenu à une de mes passions un moment délaissée : l'informatique. En effet le champ de recherche qu'il m'avait proposé alors, la Bioinformatique, m'a permis de concilier et d'imbriquer deux disciplines qui m'intéressent au plus haut point : la Biologie et l'Informatique.

De la même manière, je remercie le Pr A.J. Cozzone tout d'abord pour m'avoir accueilli au sein de l'Institut de Biologie et Chimie des Protéines (CNRS – UPR 412), mais également pour la confiance qu'il m'a accordée en me confiant la création et l'administration du serveur Web de l'EDISS, École Doctorale Interdisciplinaire Sciences-Santé, qu'il dirige et à laquelle j'ai appartenu au cours de ma formation doctorale.

Je remercie tous les membres du Laboratoire de Conformation de Protéines pour leur bonne humeur et leur disponibilité, gages d'un climat de travail de qualité: Anja Böckmann, Christophe Combet, Christophe Geourjon, Laurent Ladavière, Roland Montserret, François Penin, Tetsuya Takahashi, mais également le presque-de-la-famille "Xav. le graveur fou". Je ne peux remercier les gens de mon équipe sans remercier un autre Xavier que je n'ai pas eu la joie de connaître plus longtemps car il nous a quitté beaucoup trop rapidement. La coutume veut de dire que ce sont toujours les meilleurs qui s'en vont, parfois c'est tragiquement vrai…

Je citais tout à l'heure l'EDISS, tous mes remerciements vont également aux membres etudiants du bureau avec qui j'ai pris plaisir

à travailler : Olivier N., Marie-Caroline N.-D., Pierre H., Frédéric B., Bruno C., pour ne citer qu'eux.

Ces mêmes personnes et toute leur petite famille se retrouvent également ici remerciées au même titre que de nombreuses autres personnes que je considère comme mes amis.

Je ne saurai terminer sans adresser tous mes chaleureux remerciements à mes parents et à ma famille, mais également à mes beaux-parents et à la famille de Christelle.

Christelle, que je remercie ici de tout mon cœur pour son amour et son soutien. Sans oublier le petit bout de chou trognon que nous avons la chance d'avoir pour fille, notre petite Alice.

Christophe

Table des Matières

Table des illustrations

Abréviations

ADN	Acide DésoxyRibonucléique
ANSI	American National Standards Institute
ARN	Acide RiboNucléique
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfert Mode
CGI	Common Gateway Interface
Cgi-bin	Common Gateway Interface BINary
DNS	Domain name Service
DSSP	Dictionnary of Secondary Structure of Proteins
FTP	File Transfert Protocol
GUI	Graphic User Interface
HTML	HyperText Mark-up Language
HTTP	HyperText Transfert Protocole
IBM	International Business Machine
Internet	INTERconnected NETwork
LAN	Local Area Network
Mac	Macintosh®
MAN	Metropolitan Area Network
MIME	Multi-purpose Internet Mail Extensions
MPSA	Multiple Protein Sequence Analysis
NPS@	Network Protein Sequence Analysis
OS	Système d'exploitation (Operating System)
OSI	Open System Interconnection
PBIL	Pôle Bioinformatique Lyonnais
PC	Ordinateur individuel (Personal Computer)
Perl	Practical Extraction and Report Language (initialement PEARL)
RAM	Random Acces Memory
RENATER	RÉseau NAtional de télécommunications pour la Technologie, l'Enseignement et la Recherche
RFC	Request For Comments
RTF	Rich Text Format
SGI	Silicon Graphics Inc.
SMTP	Simple Mail Transfert Protocol
SUN Microsystem	Stanford University Network Microsystem

Table des illustrations

TCP/IP	Transfert Control Protocole - Internet Protocole
UDP	User Datagram Protocol
UNIX	UNiplexed Information and Computing Services (UNICS)
URL	Uniform ressource Locator
VIBRANT	Virtual Interface for Biological Research ANd Technology
WAN	Wide Area Network
WWW, W3, Web	World Wide Web

1 Introduction

Le repliement des protéines est un des grands challenges de la Bioinformatique : prédire la structure tridimensionnelle d'une protéine à partir de sa séquence d'amino-acides. Il est largement reconnu que cette séquence contient toute l'information nécessaire à son repliement correct, puisque celui-ci est apparemment déterminé thermodynamiquement. C'est le principe thermodynamique d'Anfinsen (P 10) énoncé en 1973. C'est de cette constatation que l'analyse de séquences de protéine tire une bonne partie de sa raison d'être. En effet, en observant et analysant la seule séquence nous devrions être capable de prédire la structure tridimensionnelle finale de la protéine ; et de là, grâce à la connaissance de sa forme et des acides aminés exposés à sa surface, accéder à ses caractéristiques physico-chimiques, aux interactions potentielles avec les autres molécules et plus globalement à la fonction qu'elle occupe dans le métabolisme de l'organisme. Même si nous en sommes encore loin, les méthodes développées nous permettent déjà d'appréhender une partie des fonctionnalités d'une protéine inconnue à partir de sa séquence en acides aminés ; et ces méthodes sont un bon prélude à la modélisation fine *de novo* du repliement d'une protéine. Cependant il apparaît maintenant clairement que cette prédiction du repliement ne sera qu'une étape à la compréhension globale des détails du fonctionnement d'une cellule et ensuite d'un organisme tout entier. Pour achever cette tâche, il sera nécessaire de connaître la majeure partie des interactions entres les molécules constituant l'organisme, c'est-à-dire des molécules de faible taille jusqu'aux macromolécules.

Depuis le début des années 1990 de nombreux programmes de séquençages intensifs ont été mis en place. Ces programmes ont pour objectif d'obtenir la séquence complète du génome d'organismes procaryotes ou eucaryotes. Depuis 1995 le rythme de publication de nouveaux génomes s'accélère au fil des mois pour atteindre dès la fin 1996 celui d'une séquence complète de génome publiée par mois. Ces différents programmes ont permis d'obtenir un panel de génomes allant des organismes procaryotes aux organismes eucaryotes. Le plus ambitieux de tous ces programmes, qui se poursuit actuellement en Europe, au Japon et aux Etats-Unis est le programme de séquençage du génome humain. Cependant la séquence de ces génomes ne serait rien sans la reconnaissance des différents gènes et l'identification de leur signification (P 44, P 136, P 32), tout comme un ouvrage imprimé tel cette thèse n'est rien qu'une suite de sigles sans la reconnaissance des mots et sans la sémantique que nous leur appliquons. De nouvelles techniques permettent d'appréhender l'organisation des génomes (P 183) ou d'identifier les mots (P 187).

La Bioinformatique est née de cette prise de conscience par les biologistes de ce que pouvait apporter l'Informatique à la Biologie pour le traitement répétitif de processus ou pour l'analyse de grandes quantités de données. Et cet apport prend toute sa mesure lors du traitement des génomes ou des grandes banques de données mondiales. Nous parlions précédemment de la reconnaissance ou de l'identification de la signification des gènes, *i.e.* de la fonction des protéines. Reconnaissance ou identification en effet puisque la nature, comme dans un gigantesque jeu de construction, utilise des briques élémentaires, les domaines protéiques, en nombre fini et les arrange de différentes manières pour obtenir un nombre plus grand de protéines différentes (P 109, P 182). De plus les protéines ainsi obtenues sont souvent utilisées dans différentes espèces pour la même fonction biologique. Et c'est lors de cette reconnaissance de caractères déjà répertoriés ou lors de l'identification de nouveaux caractères qu'intervient un domaine particulier de la Bioinformatique : l'analyse de séquence, et plus particulièrement l'analyse de séquences de protéine. Cette analyse pourra être approfondie, secondée et/ou (in)validée par l'apport de résultats expérimentaux *in vitro* et/ou *in vivo*.

Toutes ces différentes phases de l'analyse de séquences de protéine disposent de nombreux algorithmes qui leur sont dédiés. Mais se pose alors le problème des formats de données requis pour la mise en œuvre de ces algorithmes, formats le plus souvent différents d'une méthode à l'autre. De plus, certains de ces algorithmes, très complexes tels les réseaux neuronaux, sont difficiles à mettre en œuvre. Ces quelques problèmes que peuvent rencontrer le biologiste l'éloigne de son objectif qu'est l'analyse de sa ou de ses protéines.

Le premier chapitre de cette thèse propose des rappels bibliographiques. Ceux-ci concernent tout d'abord le système d'information répartie que propose le Web et son importance pour les biologistes. Ensuite sont évoquées les différentes bases de données internationales relatives aux protéines. Ce sont les banques généralistes ou spécialisées relatives aux séquences de protéines ou aux autres données biologiques, mais également les outils d'interrogation de ces banques. Puis sont abordées les méthodes d'alignement et de prédiction de structure secondaire de séquences protéiques. Les alignements peuvent être réalisés entre deux séquences pour une recherche de similarité ou entre plusieurs séquences pour l'étude de cette famille de protéines. Ce premier chapitre se termine par une analyse des outils bioinformatiques existants.

Le deuxième chapitre traite des différents matériels et méthodes utilisés pour la réalisation de ces travaux. Il évoque notamment l'hétérogénéité des architectures matérielles et logicielles des systèmes existants (UNIX, Macintosh et Windows), ainsi que les protocoles réseaux utilisés pour relier ces systèmes répartis (TCP/IP, HTTP, MIME type, …). Les langages employés pour les différentes réalisations sont abordés (langage C, Perl, PostScript, RTF, HTML) et la méthodologie employée pour le développement de logiciel en langage C est décrite de façon plus détaillée. Le dernier paragraphe traite de l'incorporation d'outils biologiques existants que sont la bibliothèque Vibrant du NCBI et ReadSeq.

Le troisième et dernier chapitre du développement de cette thèse détaille les réalisations effectuées au cours de ces travaux. La recherche de motifs protéiques dans les banques de séquences est un outil essentiel pour identifier certaines familles de protéines. Souvent il est nécessaire de rechercher ces motifs en autorisant une marge d'erreur. Il n'existe pas de méthode de recherche de motifs permettant de limiter les erreurs sur les positions les plus strictes et de les privilégier par rapport aux positions les plus dégénérées. C'est pourquoi nous avons développé une méthode qui permet de privilégier la conservation des positions les plus strictes lors d'une recherche d'un motif avec un taux d'erreur donné. Ces positions les plus strictes sont celles qui sont biologiquement les plus significatives du motif utilisé. Cette méthode augmente le rapport signal sur bruit lors d'une recherche. Cette méthode est disponible dans différentes réalisations logicielles abordées dans le paragraphe suivant : ProScan, PattInProt, *biolcp*, NPS@ et MPSA.

L'écriture de code source réutilisable est une nécessité afin de pouvoir l'intégrer directement lors d'utilisations ultérieures. Dans ce contexte, nous avons écrit la bibliothèque *biolcp* qui regroupe différentes méthodes et des fonctionnalités client-serveur appliquées à l'analyse de séquences protéiques. Cette bibliothèque est disponible sur le Web et ses outils peuvent être utilisés par la communauté scientifique internationale pour la réalisation d'autres projets. NPS@ est un serveur Web dédié à l'analyse de séquences de protéine. Il propose aux biologistes de nombreux outils interconnectés et des liens sur les banques de données internationales. NPS@ répond ainsi à la nécessité de combiner plusieurs méthodes pour conduire une analyse de séquences, mais également de disposer d'une interface indépendante du système d'exploitation. Mais les possibilités du Web restent limitées pour l'interactivité, de par sa conception basée sur le langage HTML et ce malgré la mise en place des nouveaux langages tels Java (URL 25) et XML (URL 57).

Aujourd'hui de nombreux programmes d'analyse de séquence existent (Chapitre 2.5). Mais aucun ne propose toutes les fonctionnalités suivantes : fonctionnement sur tous les systèmes d'exploitation, incorporation de prédictions de structures secondaires et de profils physico-chimiques dans un alignement multiple de protéines, graphes interactifs et couplés à l'alignement multiple, fonctionnalités client-serveur. C'est pourquoi nous avons développé MPSA. En effet MPSA intègre de nombreuses méthodes d'analyses et répond au problème soulevé par l'échange de données entres ces méthodes qui requièrent des formats différents. Les résultats obtenus avec ses méthodes subissent le plus souvent une pré-analyse automatique matérialisée par différents codes. De plus, de par son implémentation en langage C et avec des éléments portables, MPSA obtient de bonnes performances lors de l'analyse de grosses quantités de données. Ce qui est nécessaire avec la mise à disposition des génomes complets et des banques internationales. Pour la même raison, MPSA fonctionne sur la majorité des systèmes d'exploitation existants. Avec l'incorporation de fonctionnalités client-serveur Web, MPSA peut accéder aux systèmes d'information répartie disponibles à l'échelle mondiale. Il s'inscrit ainsi dans une politique de gestion centralisée des données avec tout de même de bonnes fonctionnalités propres afin de pouvoir conserver une certaine autonomie.

2 Rappels bibliographiques

L'analyse de séquences de protéine consiste à extraire de l'enchainement des acides aminés le maximum d'informations sur les caractéristiques physico-chimiques, structurales et fonctionnelles de la protéine considérée. Pendant longtemps l'information à la disposition des biologistes était limitée à celle obtenue grâce aux congrès, publications et collaborations. Avec l'avénement d'Internet et l'évolution exponentielle des matériels informatiques, de nombreux outils sont à la disposition du biologiste.

De nos jours, l'informatique et les réseaux permettent un bon niveau d'analyse des séquences. C'est pourquoi le premier réflexe d'un biologiste est de chercher à obtenir le maximum de la séquence du gène qu'il étudie. Il traduit ensuite cette séquence de nucléotides en séquence d'acides aminés. A l'aide de sa séquence d'acides aminés il va prédire, ou obtenir par homologie ou analogie, de nombreuses caractéristiques de cette protéine, voire même sa structure et sa fonction. Puis il confirmera les résultats de l'analyse par des tests biochimiques ou de biologie moléculaire. En effet l'analyse de séquence n'a de sens que par l'aide qu'elle fournit aux biologistes pour progresser beaucoup plus rapidement dans l'étude des protéines d'un organisme.

Les techniques bioinformatiques d'analyse disponibles sont tout d'abord les banques de données, fondement essentiel sur lequel s'appuie la majorité de méthodes d'analyse. Pour travailler avec ces banques de données, il existe de nombreux outils d'interrogation concernant par exemple la recherche de similarité ou la recherche de sites et signatures fonctionnelles. Les méthodes d'alignement multiple permettent de travailler sur un groupe de protéines homologues, appartenant à la même famille, ou similaires, possédant des caractéristiques communes. Un outil essentiel pour accéder à la structure d'une protéine est la prédiction de structure secondaire de protéine, elle permet de mettre en évidence des similarités entre des protéines, similarités passées inaperçues lors de l'étude de la séquence d'acides aminés ; c'est également une première étape de l'étude structurale de la protéine considérée. D'autres méthodes telles la prédiction de caractéristiques physico-chimiques, la reconnaissance des sections transmembranaires ou des zones enfouies et exposées sont également très employées lors de l'analyse d'une séquence de protéine.

Toutes ces méthodes et leur automatisation sont un premier jalon nécessaire à la compréhension de l'organisation moléculaire des êtres vivants. Celle-ci passe par une analyse des séquences complètes de génome disponibles, et surtout par la cartographie des chemins métaboliques et des interactions entre les protéines de l'organisme. Car le séquençage de génome complet n'a d'intérêt que pour l'étude approfondie des protéines traduites d'après ces gènes, c'est-à-dire l'étude des fonctions de ces protéines et de leurs interactions mutuelles.

2.1 Le Web : le nouvel outil du biologiste

Nous ne pouvons aller plus loin sans dire quelques mots sur la révolution qui se déroule depuis quelques années dans le domaine de l'information. Cette révolution est Internet et sa composante la plus connue du grand public : le Web. En effet plus qu'Internet, c'est surtout le World-Wide Web qui a provoqué une véritable révolution dans la mise à disposition des informations et leur accessibilité à l'échelle mondiale.

Au début des années soixante-dix les informaticiens ont assemblé le premier réseau reliant plusieurs machines situées en des lieux différents : ce fut la naissance du réseau ARPAnet, en 1969, mis en place par la DARPA (Defense Advanced Research Project Agency) afin de faciliter l'accès aux

gros systèmes distants. Ce réseau ne devait pas avoir de points névralgiques : aucun nœud de ce réseau n'était nécessaire à la transition des données, celles-ci prenaient des chemins différents suivant les circonstances du moment de leur emission. À partir de 1977, le modèle TCP/IP connut un grand succès et fut mis en œuvre à l'échelle mondiale puisque de nombreux réseaux sous TCP/IP virent le jour durant les années soixante-dix et quatre-vingts. Tous ces réseaux furent connectés entre eux, et en 1983 la composante militaire du réseau[1] se sépara d'une partie publique qui devint l'Internet tel que nous le connaissons.

Les recherches en la matière ont profité à toute la communauté scientifique, et notamment à la communauté biologique puisque dès le milieu des années quatre-vingts les premières banques de données biologiques sont apparues sur Internet (P 76, P 16). Ce furent les prémices d'une discipline nouvelle : la Bioinformatique.

Cette notion de banques de données centralisatrices accessibles à travers un réseau informatique fut une des premières étapes du processus qui allait aboutir à l'analyse de séquence moderne. Au début, peu de laboratoires de biologie étaient connectés à Internet. L'utilisation des ces banques restait l'appanage d'un petit nombre de laboratoires qui utilisaient ces données pour mettre au point des méthodes informatiques d'analyse de séquences. Mais ces méthodes ne fonctionnaient pour la plupart que sur des ordinateurs utilisant UNIX ou VMS comme système d'exploitation et l'accès à Internet passait par l'utilisation de programmes aux noms exotiques tels « telnet », « gopher » ou « FTP ». Il fallait avoir des notions en Informatique très poussées pour pouvoir installer et utiliser ses méthodes ou accéder à Internet.

Une véritable révolution des techniques de l'information allait se produire au début des années quatre-vingt-dix. En 1989 Tim Berners-Lee posait les premiers jalons du « World-Wide Web », le tissu planétaire d'information (P 24). Ce concept fut développé au CERN (URL 9) situé à proximité de Genève. Le Web offre une méthode très conviviale pour mettre des données sur le réseau. Il s'appuie sur un protocole, HTTP, qui se place au-dessus de TCP/IP. Mais le trait de génie réside dans le langage HTML, HyperText Mark-up Language. Le principe repose sur la présence dans le texte courant de « liens hypertextes ». Ce sont un mot ou un groupe de mots « pointant » sur un autre document : un simple clic sur ce lien hypertexte provoque l'affichage du document auquel il était relié. Le langage HTML propose également une mise en page très simple et performante des données au format ASCII. Toutes ces fonctionnalités, le mode hypertexte et la mise en page, sont accessibles à l'aide d'une syntaxe très simple employant diverses balises (Tableau 13). Ainsi des documents très complexes sont codés par un simple fichier ASCII, ce qui rend le langage HTML indépendant des architectures informatiques, *i.e.* le « hardware ».

La puissance et les potentialités de ce concept sont telles que le Web connaît depuis le milieu des années quatre-vingt-dix un succès énorme auprès du grand public. Le Web permet en effet d'accéder à l'information de façon simple et efficace sur une échelle planétaire. Et cet engouement est d'autant plus fort que les industriels se sont impliqués par des investissements massifs car ils se sont rapidement rendus compte des potentialités du Web pour un nouveau médium économique : le commerce électronique. Et cette implication des industriels est bienvenue puisqu'elle permet la maintenance du réseau et parfois de soutenir[2] des initiatives essentielles de la communauté biologique

[1] actuellement le réseau *milnet*
[2] À propos du soutien de SWISS-PROT : http://www.expasy.ch/announce, http://www.ebi.ac.uk/news.html

comme ce fut le cas pour la banque de séquences de protéine SWISS–PROT en septembre 1998 (P 16).

Les biologistes se sont, eux aussi, rapidement rendus compte du formidable outil que pouvait être le Web (P 97, P 178). De nombreux sites existent de par le monde comme le site Deambulum[3] sur le serveur national InfoBioGen (URL 23) et la page des liens WWW d'Amos Bairoch[4]. Tous les sites accessibles proposent soit des bases de données soit des algorithmes originaux qui peuvent être mis en œuvre simplement.

2.2 Les bases de données relatives aux protéines

L'information la plus complète et pertinente sur les protéines peut être trouvée dans les banques de données disponibles sur Internet. Cet accès facile grâce à une interface Web est doublé d'une grande expertise dans la présentation et la gestion des données puisque les premières banques de données relatives aux protéines sont apparues sur Internet dès le début des années quatre-vingts avec la banque PIR (P 76). Depuis quelques années la quasi totalité des banques est disponible sur Internet (Tableau 1, Chap. 2.2.1). Toutes ces banques sont accessibles à travers une interface Web, ce qui présente le double avantage de permettre une mise à jour efficace des données et une interconnexion forte des différentes banques par le jeu de références croisées utilisant des liens hypertextes (P 2, P 4, P 3).

Tableau 1 : **Quelques unes des principales banques de données biologiques.**

Banques	**Description**	**Mise à jour**	**Interface Web**
GenBank	Séquences nucléiques	quotidienne	http://www.ncbi.nlm.nih.gov/Web/Search/index.html
EMBL	Séquences nucléiques	quotidienne	http://www.ebi.ac.uk/ebi_docs/embl_db/ebi/topembl.html
SWISS–PROT	Séquences protéiques	hebdomadaire	http://www.expasy.ch/sprot/
TrEMBL	Séquences protéiques	hebdomadaire	http://www.ebi.ac.uk/ebi_docs/swissprot_db/retreival.html
PIR-International	Séquences protéiques	hebdomadaire	http://www-nbrf.georgetown.edu/pir/searchdb.html
PDB	Structures tridimension-nelles de protéine	quotidienne	http://www.pdb.bnl.gov jusqu'au 30 juin 1999. http://www.rcsb.org à partir du 1er juillet 1999
EST	Banque d'étiquettes	quotidienne	http://www.ncbi.nlm.wnih.gov/dbEST/index.html

Elles sont également pour la grande majorité disponibles sous la forme d'un fichier texte simple, *i.e.* d'un fichier ASCII, récupérable par l'intermédiaire d'une requête FTP. Le format ASCII a l'avantage d'être visualisable à l'aide de n'importe quel éditeur de texte, et d'être aisément utilisable dans le cadre de la conception de nouvelles méthodes d'analyse (P 156). Cependant une mise en forme est présente afin de délimiter les différentes entrées, et au sein d'une même entrée afin de délimiter différents champs tels le numéro d'identification, la séquence, les commentaires, les références bibliographiques, *etc*. Ces mises en formes sont différentes suivant les banques de données et cette disparité des formats pose le problème d'une accession universelle à l'information contenue dans ces banques de données biologiques.

On peut distinguer les banques dites généralistes (SWISS–PROT, TrEMBL, PIR, *etc.*) des banques spécialisées (YPD, PROSITE, *etc.*). Les premières ont pour vocation d'être les plus exhaustives possibles, *i.e*. de rassembler la totalité des séquences ou informations connues pour toutes

[3] InfoBioGen, le Deambulum : http://www.infobiogen.fr/services/deambulum/fr
[4] http://www.expasy.ch/alinks.html

les protéines de l'ensemble des espèces avec ou sans expertise particulière. Les secondes se sont constituées autour de thématiques biologiques, afin de réunir les séquences d'une même espèce ou les séquences d'une famille protéique pour toutes les espèces. Elles dérivent le plus souvent des banques généralistes et nécessitent l'intervention d'experts qui sont la plupart du temps les auteurs de la banque.

2.2.1 DBcat : la banque des banques de données biologiques

Le serveur national français InfoBioGen (URL 23) héberge et maintient la banque DBcat (P 59, URL 11). DBcat est un catalogue exhaustif et détaillé de toutes les banques de données biologiques. En effet pour des raisons aussi bien scientifiques, qu'historiques et politiques les données biologiques sont réparties en de nombreuses banques de données indépendantes. Cette banque des banques de données contient 412 entrées (mars 1999) réparties en 8 domaines (Tableau 2). Chaque entrée référencée, *i.e.* chaque banque de données biologiques, contient différents champs indiquant, entre autres, une brève description de la banque, le domaine auquel elle appartient, le nom des auteurs, la référence bibliographique à citer, les adresses Web, ftp et postale.

Tableau 2 : Les différents domaines du catalogue de banques de données biologiques DBcat.

Domaine	Nombre d'entrées
ADN	60
ARN	22
Protéine	75
Génome	58
Cartographie	29
Structure de protéine	18
Littérature	37
Divers	113
Total	412

2.2.2 Les banques de séquences de protéines généralistes

2.2.2.1 SWISS–PROT

La banque SWISS–PROT (P 16, URL 51) a été créée en 1986 par Amos Bairoch au sein du département de Biochimie Médicale à l'Université de Genève. A partir de 1988 elle fut maintenue en collaboration avec l'EMBL (URL 16). Elle est maintenant le fruit d'un partenariat entre l'Institut Suisse de Bioinformatique (SIB, URL 48) et l'Institut Européen de Bioinformatique (EBI, URL 14), une annexe de l'EMBL basée à Hinxton Hall.

La banque SWISS–PROT se définit elle-même comme différente des autres banques de séquences de protéine par une annotation de chaque séquence, une faible redondance et une forte interconnexion avec les autres banques.

Chaque entrée de SWISS–PROT, *i.e.* chaque bloc de données relatives à une séquence de protéine, est composée d'un « cœur » des données (la séquence, les références bibliographiques et les données taxonomiques) et des annotations. Celles-ci indiquent entre autre la ou les fonctions de la protéine, les modifications post-traductionnelles, les domaines et les sites fonctionnels, les structures secondaires et quaternaires, les similarités avec les autres protéines. Ces données sont très importantes

lors du processus d'analyse de séquences et sont très utiles au biologiste. Elles sont insérées de différentes manières par les auteurs ou par un groupe d'experts extérieurs.

Une redondance minimale est obtenue en fusionnant toutes les données relatives à une même séquence étudiée par plusieurs auteurs. L'interconnexion avec les autres banques est réalisée par des références croisées (le plus souvent un numéro d'accession dans la banque considérée) avec les banques GenBank (ADN, P 23), PROSITE (sites et signatures fonctionnelles, Chap. 2.2.4.1.1), Medline (littérature, URL 27), *etc*. Actuellement SWISS–PROT intègre des références à 29 banques de données différentes.

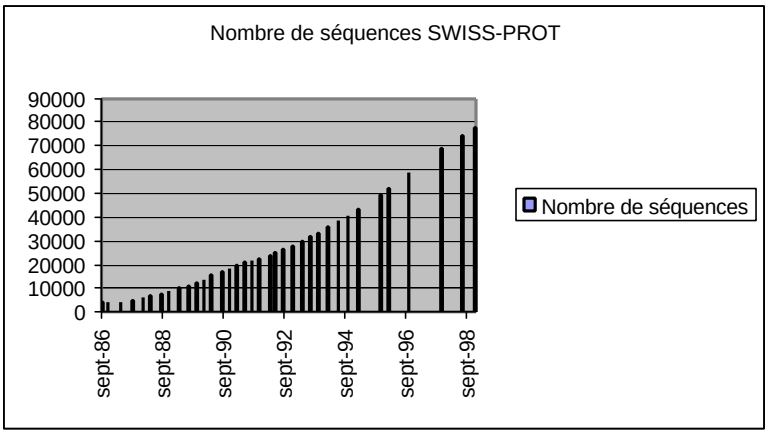


Figure 1 : Croissance de la banque SWISS–PROT depuis sa création en 1986.

Depuis sa création en 1986, la croissance de la banque SWISS–PROT suit une courbe exponentielle (Figure 1). La durée de réalisation d'une nouvelle version a augmenté ces dernières années. Dans sa dernière version (version 37 de décembre 1998) elle contient 77 977 entrées protéiques annotées, soit 28 268 293 acides aminés. Toutes ces séquences proviennent de plus de 6 000 espèces différentes même si les 10 espèces les plus représentées totalisent à elles seules près de la moitié des entrées (Tableau 3). Certaines espèces ont été sélectionnées pour servir d'organismes modèles du fait des programmes de séquençage dont elles font l'objet (Tableau 3).

Tableau 3 : Les organismes modèles de la banque SWISS–PROT (P 16)

Organisme	**Banque de donnée croisée**	**Fichier index**	**Nombre de séquences**
H.sapiens	MIM	MIMTOSP.TXT	5146
S.cerevisiae	SGD	YEAST.TXT	4806
E.coli	EcoGene	ECOLI.TXT	4476
M.musculus	MGD	MGDTOSP.TXT	3387
B.subtilis	SubtiList	SUBTILIS.TXT	2046
C.elegans	Wormpep	CELEGANS.TXT	1956
H.influenzae	HiDB (TIGR)	HAEINFLU.TXT	1701
S.pombe	None yet	POMBE.TXT	1406
M.jannaschii	MjDB (TIGR)	MJANNASC.TXT	1307
D.melanogaster	FlyBase	FLY.TXT	1064
M.tuberculosis	None yet	None yet	918
A.thaliana	None yet	In preparation	792

Organisme	Banque de donnée croisée	Fichier index	Nombre de séquences
S.typhimurium	StyGene	SALTY.TXT	723
M.genitalium	MgDB (TIGR)	MGENITAL.TXT	470
H.pylori	HpDB (TIGR)	HPYLORI.TXT	367
D.discoideum	DictyDB	DICTY.TXT	285
C.albicans	None yet	CALBICAN.TXT	194
S.solfataricus	None yet	None yet	84

2.2.2.2 TrEMBL

La banque de séquences de protéine TrEMBL est née en 1996 d'une double constatation paradoxale (P 16) : d'un côté les projets génomes dévoilent de nouvelles séquences et ce avec un rythme croissant, d'un autre côté SWISS–PROT doit conserver les qualités qui ont fait sa renommée, une annotation de grande qualité et une faible redondance. En effet, les nouvelles séquences obtenues par les projets génomes doivent être rapidement incorporées à SWISS–PROT pour que celle-ci continue de refléter au mieux l'état des connaissances. Cependant l'annotation de ces nouvelles séquences doit être d'aussi bonne qualité que les autres entrées, et il clair que ce deviendrait rapidement impossible car la vitesse d'annotation de SWISS–PROT atteindrait rapidement un palier : l'annotation étant réalisée manuellement par les auteurs et un groupe d'experts internationaux. C'est pour résoudre ce paradoxe que TrEMBL, « translation of EMBL nucleotide sequence database », fut créée en 1996 comme une antichambre à SWISS–PROT (Figure 2).

TrEMBL est constituée des séquences protéiques issues de la traduction des séquences codantes (CDS) de la banque de séquences nucléotidiques de l'EMBL (P 175, URL 15). La version 7 de TrEMBL a été publiée en août 1998 et contenait la traduction des 327 000 séquences codantes de la banque de l'EMBL version 55. Parmi celles-ci, environ 109 000 CDS sont déjà répertoriées dans SWISS–PROT, elles ne sont donc pas retenues. Il en reste donc environ 218 000. De ces dernières sont éliminées les redondances, ce qui ramène le nombre d'entrée de TrEMBL à 193 860.

La banque TrEMBL est séparée en deux sous-ensembles de séquences suivant leur future incorporation à SWISS–PROT : SP–TrEMBL et REM–TrEMBL (Figure 2). Les séquences de SP–TrEMBL (SWISS–PROT–TrEMBL) seront incorporées à la prochaine version de SWISS–PROT et possèdent déjà un identifiant SWISS–PROT, cela concerne 165 420 séquences de TrEMBL version 7. Les autres séquences, 28 440, forment REM–TrEMBL, « REMaining TrEMBL ». Elles ne seront pas retenues dans SWISS–PROT car elles appartiennent à l'une des cinq catégories suivantes :

- séquences de récepteur de cellule T ou d'immunoglobuline,
- séquences synthétiques,
- fragments de moins de 8 acides aminés,
- séquences issues de brevets,
- traduction de CDS se révélant, avec certitude, comme ne codant pas de véritable protéine.

Ce n'est donc que le sous-ensemble SP–TrEMBL qui peut être considéré comme l'antichambre de SWISS–PROT. Parmi les 165 420 séquences, pour TrEMBl version 7, il y en a environ 40 000 qui ne sont que des informations supplémentaires à des protéines déjà répertoriées dans SWISS–PROT. Les séquences de SP–TrEMBL subissent deux étapes avant d'être incorporées à

SWISS–PROT. La première étape consiste à réduire les redondances en fusionnant toutes les entrées relatives à une même protéine, à l'aide d'outils comme LASSAP (P 84). La seconde étape est une augmentation de l'information sur la séquence notamment à l'aide de la banque de motifs PROSITE (P 105) et de l'algorithme *eMotif* (P 142), ainsi que de la banque de donnée ENZYME (P 15).

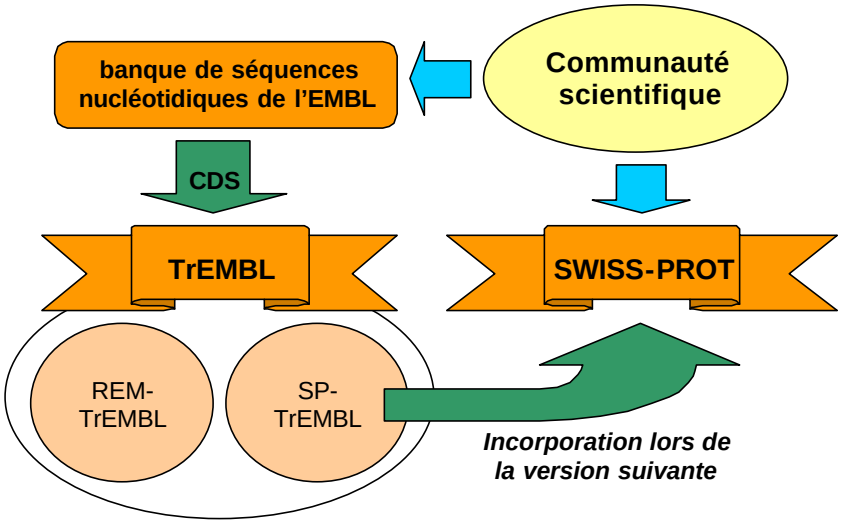


Figure 2 : Relations entre les banques de séquences de protéine SWISS-PROT et TrEMBL

2.2.2.3 PIR International

La banque de séquences de protéine PIR, Protein Information Ressource (P 20, P 77), est une des plus anciennes banques de données biologiques puisqu'elle est disponible sur Internet depuis 1984 (P 76). Elle fut mise en place par le National Biomedical Research Foundation (NBRF) comme succession à l'« Atlas of Protein Sequence and Structure » maintenu pendant 20 ans par Margaret O. Dayhoff (P 47). Depuis 1988, la banque PIR-International fait l'objet d'une collaboration internationale entre le NBRF, le Munich Information Center for Protein Sequence (MIPS, URL 28) et la Japan International Protein Information Database (JIPID).

PIR est une banque de séquences de protéine annotées, non-redondante et organisée en super-familles de protéines (P 46, P 49, P 21) et domaines d'homologie (P 20). Chaque entrée contient de nombreuses références croisées avec les banques GenBank, EMBL, DDBJ[5], PDB, Medline, entre autres. Cette banque est divisée en quatre sous-ensembles : PIR1, PIR2, PIR3 et PIR4. PIR1 et PIR2 contiennent 99% des entrées de PIR. PIR1 ne contient que des séquences fortement annotées, non-redondantes et classées en super-familles de protéines. Les séquences répertoriées dans PIR2 sont pour la majorité d'aussi bonne qualité que celles de PIR1. PIR3 sert de tampon d'entrée : elle contient des séquences récemment intégrées à PIR et qui n'ont pas reçu l'expertise apportée aux entrées répertoriées dans PIR1 et PIR2. Le sous-ensemble PIR4 contient, quant à lui, les séquences « non-naturelles » : les pseudo-gènes, les phases ouvertes de lecture (ORF[6]) non-exprimées, les séquences synthétiques, *etc*. Les séquences de la banque PIR ont été alignées en regroupant les séquences par super-famille, famille et domaine d'homologie. Les alignements obtenus à l'aide de Clustal V/W (P 180) sont regroupés dans la banque PIR-ALN (P 174).

[5] DDBJ : Dna Data Bank of Japan.
[6] ORF : Open Reading Frame.

Le MIPS propose différentes banques de données biologiques telles PEDANT[7] (P 134, URL 38) qui propose une analyse systématique et à jour des séquences de génome publiés, MYGD[8] (URL 28) une banque complète du génome de la levure.

2.2.3 Les banques de séquences de protéine spécialisées

Du fait de l'augmentation exponentielle du nombre de séquences de protéine publiées, la nécessité de développer des banques spécialisées est rapidement apparue. Ainsi aujourd'hui, il en existe de nombreuses ; les informations disponibles et leur mise à jour sont très variables. Ces bases de données sont généralement dédiées à une famille de protéines. On peut par exemple citer :

- YPD (P 103, URL 58) et SGD (P 37, URL 47) sont dédiées à *Saccharomyces cerevisiae*,
- GCRDb, une banque sur les récepteurs couplés aux protéines G (P 118, URL 19).
- IMGT, la banque spécialisée dans les immunoglobulines, les récepteurs des lymphocytes T et le complexe majeur d'histocompatibilité (P 123, URL 22).

Une liste très complète est maintenue par Amos Bairoch à l'adresse suivante : http://WWW.expasy.ch/amos_www_links.html#Gene_prot.

2.2.4 Autres bases de données biologiques utiles en analyse de séquences

D'autres banques n'ont pas pour vocation de regrouper des séquences de protéine, mais recensent toutes les informations biologiques sur un thème particulier, par exemple :

- les banques bibliographiques : Medline (URL 27) et SeqAnalRef (P 14, URL 46),
- la banque KEGG (P 145, URL 26) décrit avec détail de nombreuses voies métaboliques d'organismes vivants, en incluant de très nombreuses références croisées sur les substrats comme sur les enzymes,
- la banque OMIM (P 132, URL 33) rassemble des informations sur les maladies génétiques humaines,
- la banque ENZYME (P 15, URL 17) répertorie les informations sur la nomenclature de 3704 enzymes.

Certaines ont une importance toute particulière : les banques regroupant des sites et signatures fonctionnelles et la Protein Data Bank (PDB).

2.2.4.1 Banques de motifs protéiques

Dans certains cas la séquence de la protéine inconnue est trop éloignée de toutes celles déjà identifiées pour qu'il soit possible de détecter une ressemblance à l'aide des outils de recherche de similarité. Cependant une relation peut être mise en évidence par l'existence d'une même disposition de certains acides aminés dans la séquence, on parle alors de la mise en évidence d'un motif ou d'une signature fonctionnelle de la famille de protéines considérée. Ces signatures protéiques sont limitées à quelques résidus, voire quelques dizaines de résidus, mais elle possède néanmoins une probabilité d'occurrence au hasard très faible, d'où leur spécificité. En effet, les acides aminés fortement impliqués

[7] PEDANT : Protein Extraction, Description and Analysis Tool.
[8] MYGD : Mips Yeast Genome Database.

dans la fonction ou la structure de la protéine subissent une pression de sélection très importante. Ils font moins souvent l'objet de mutation, ou alors ces mutations sont conservatrices (P 48, P 50).

2.2.4.1.1 PROSITE

Ainsi depuis maintenant une dizaine d'année plusieurs banques de données spécialisées ont été développées. La plus connue est PROSITE (P 105), elle décrit actuellement (mars 1999) plus de 1 000 familles ou domaines protéiques. Certains d'entre eux sont caractérisés par l'existence de plusieurs motifs ou signatures. Même si cette recherche de site est très performante, certaines familles, du fait d'une très forte divergence de séquences, possèdent des domaines structuraux ou fonctionnels qui ne peuvent être exprimés avec suffisamment de pertinence en utilisant les motifs. Dans ces cas, une solution peut être apportée par l'utilisation des techniques dites de "profils" qui utilisent une matrice des occurrences de chaque acide aminé à chaque position du motif.

2.2.4.1.2 PRODOM, DOMO et Pfam

Les banques PRODOM (P 41, P 42, URL 43), DOMO (P 89, P 90, URL 13) et Pfam (P 22, URL 40) décrivent les domaines que l'on peut rencontrer à l'intérieur des différentes familles de protéines en utilisant un critère de similarité de séquences. Ces banques proposent les alignements multiples utilisés pour la définition de ces domaines.

Pfam contient également les profils HMM[9] de ces domaines. Les protéines utilisées pour la construction de cette banque sont celles de SWISS-PROT et SP-TrEMBL, et les familles de protéines sont celles identifiées par PROSITE. PRODOM utilise également les séquences des protéines référencées dans SWISS-PROT. Les similarités de séquences sont détectées avec l'outil PSI-BLAST (Chapitre 2.3.2.2). Par contre, DOMO utilise un sous-ensemble des protéines non-redondantes de SWISS-PROT et de PIR. La délimitation des domaines dans ces protéines est réalisée par une recherche de similarités locales suivie de l'alignement multiple des séquences retenues. DOMO contient un plus grand nombre de domaines que PRODOM et les familles sont plus peuplées. De plus DOMO présente un taux d'erreur relativement faible (6,8%) par rapport à PRODOM (34,9%) alors qu'elles sont toutes les deux construites de façon automatisée, et ce taux n'est pas très différent de celui de PROSITE (3,4%), construite manuellement.

2.2.4.1.3 BLOCKS et PRINTS

A partir des listes des protéines membres d'une famille contenues dans PROSITE, Henikoff *et al.* (P 100) ont développé la base de données BLOCKS (URL 7) rassemblant les alignements multiples sans insertion correspondant aux régions les mieux conservées de familles de protéines documentées.

PRINTS est une banque de données qui contient des « empreintes » (« fingerprints ») de protéines (P 13, URL 42). Ces empreintes consistent en un regroupement de plusieurs motifs afin de définir une signature de familles de protéines (P 151, P 12). La version 20.0 de septembre 1998 contient 990 empreintes de protéines construites à partir d'environ 5 700 motifs.

[9] HMM : Hidden Markov Model

2.2.4.2 Banque de structures tridimensionnelles : la PDB

La connaissance de la structure tridimensionnelle d'une protéine est cruciale pour une meilleure compréhension de son activité et pour pouvoir envisager des applications de « drug design » ou de « docking ». Les techniques disponibles pour cette détermination de la structure d'une protéine sont la cristallographie aux rayons X (P 27) et la spectroscopie par résonance magnétique nucléaire (RMN, P 186). Cependant, cette information est très difficile à obtenir et nécessite un investissement très important. Ceci explique le décalage important entre le nombre de structures tridimensionnelles connues et le nombre de séquences de protéine connues.

L'ensemble des données tridimensionnelles relatives aux macromolécules biologiques est rassemblé dans la banque PDB (P 1, P 25, URL 37). Dans la version 87 de janvier 1999 elle contient 9179 entrées (8143 protéines ou peptides, 381 complexes protéines/acides nucléiques, 643 acides nucléiques et 12 carbohydrates). Pourtant le nombre de protéines dont la structure tridimensionnelle est connue est inférieur à 8143. En effet, une même protéine peut posséder dans la banque PDB plusieurs entrées, voire même dans certains cas plusieurs dizaines d'entrées, du fait de conditions expérimentales différentes (mutagenèse dirigée, présence d'effecteur, résolutions différentes de la structure, utilisation de techniques d'études structurales différentes).

En ne considérant que les chaînes protéiques présentant moins de 95% d'identité entre elles, le nombre de structures disponibles n'est que de 2947, à 25% d'identité seules 1028 structures subsistent sur les 8143 entrées répertoriées. Ces sous-ensembles de la PDB sont disponibles de 25% à 90% d'identité par paire avec un incrément de 5% (P 102). Le seuil de 25% d'identité dans un alignement par paire pour plus de 80 résidus est le seuil de confiance pour la modélisation moléculaire par homologie : il est considéré que ce sont les conditions minimales à remplir par deux séquences pour garantir des similarités de structure (P 169).

Il existe de nombreux logiciels académiques permettant de visualiser une structure tridimensionnelle de macromolécules biologiques à partir d'une entrée PDB : RasMol (P 170, URL 44), Swiss-Pdb Viewer (P 95, URL 50), AnTheProt (P 80, URL 2),*etc*.

2.2.4.3 Banque de domaines structuraux : SCOP, CATH, *etc.*

De nombreuses banques utilisent les données de la PDB pour classer les protéines suivant leurs domaines structuraux : par exemple les banques SCOP ou CATH. La banque SCOP (P 138, P 108, URL 45) répartit les structures connues de protéine en différentes catégories suivant la proximité phylogénétique (famille et superfamille) des séquences et la présence de motifs structuraux communs (repliement commun et classe). La banque CATH (P 146, P 147, URL 8) regroupe les protéines d'après les structures secondaires présentes (classe), l'arrangement conformationnel (architecture) ou séquentiel (topologie) de ces structures secondaires et au plus haut niveau (superfamille homologue) les protéines d'une même topologie ayant une forte similarité de séquences et de fonction.

2.2.5 Outils d'interrogation de bases de données

Jusqu'à présent un effort important portait sur l'acquisition des données biologiques relatives aux protéines. Le résultat est probant et les progrès de l'automatisation des techniques de biochimie et de biologie moléculaire sont tels que l'information relative aux protéines croit à une vitesse exponentielle, notamment si on considère les programmes génomes en cours et à venir. Maintenant une priorité est donnée à la mise au point de techniques rapides et efficaces d'analyse des données en accordant une attention particulière à la pertinence des résultats.

Il existe plusieurs systèmes d'interrogation de bases de données. Beaucoup bénéficient d'une interface Web et permettent des interrogations avec plusieurs critères sur plusieurs banques de données. Deux des systèmes d'interrogation des banques de données sont SRS (P 64, P 63, URL 49) et ACNUC (P 88, URL 1). Tous deux permettent l'utilisation simultanée de plusieurs critères de sélection (mots-clefs, date, auteurs, espèce, code d'accès dans la banque, nom du gène, *etc.*) en utilisant les opérateurs logiques ET, OU et NON. Il est ainsi possible d'extraire facilement la totalité des séquences protéiques de kinase humaine dont on connaît le gène et ayant été séquencées en 1998. Il est ensuite possible d'étudier les différentes séquences obtenues en les transférant à des algorithmes d'analyse de séquence (recherche de similarités dans les banques, alignement multiple, prédiction de structure secondaire, *etc.*) comme c'est le cas avec ACNUC sur le site du PBIL (URL 35).

SRS est le système d'interrogation de bases de données actuellement le plus performant et le plus polyvalent. Il est fondé sur une conception et un langage orientés objet. Chaque base de données est un objet sur lequel peuvent être appliquées de nombreuses fonctions de manipulation, et ce quel que soit le format des données. Les banques de données gérées par SRS doivent simplement être sous la forme d'un fichier texte ASCII. Ainsi, ce peuvent être par exemple la PDB, Medline, PROSITE, ou n'importe laquelle des banques de données de séquences. SRS est disponible sur de nombreux sites « miroirs », *i.e.* identique au site principal (URL 49), par exemple celui hébergé par le serveur national InfoBioGen : http://www.infobiogen.fr/srs

2.3 Alignement de séquences de protéine

L'alignement de séquences de protéine consiste à faire correspondre les portions communes de ces séquences. Un alignement multiple est commodément représenté par une matrice où chaque séquence représente une ligne. L'alignement fait correspondre les acides aminés conservés (un seul acide aminé présents dans la colonne considérée), les mutations conservatives (les acides aminés présent dans la colonne sont similaires d'après des critères de taille, de polarité, d'hydrophobie ou de réactivité chimique) et les régions divergentes (les acides aminés présents dans la colonne n'ont pas de caractéristique commune).

L'alignement multiple matérialise alors les processus évolutifs potentiels qui ont conduit à ces séquences à partir d'une séquence ancestrale. Ces processus affectant la séquence d'acides aminés sont de trois types : insertion, suppression et mutation d'acides aminés.

2.3.1 Algorithme de programmation dynamique de Smith et Waterman

Les premiers développements tendant à identifier des portions communes à plusieurs séquences ont été réalisés par Needleman et Wunsch (P 140). En utilisant une méthode par calcul itératif de matrices. De nombreux autres algorithmes heuristiques ont été publiés par la suite jusqu'à ce qu'en 1981 Smith et Waterman (P 173) proposent une méthode d'identification de sous-séquences communes qui allait devenir l'un des principaux piliers de l'alignement de séquences. Cette méthode dite de « programmation dynamique » permet d'identifier une paire de segments communs à deux séquences de telle manière qu'il n'y ait pas d'autre paire avec un taux de similarité plus fort. Elle permet par extension d'identifier et de classer suivant le taux de similarité tous les segments communs à deux séquences, et en reliant les principaux segments d'obtenir un alignement optimal entre les deux séquences.

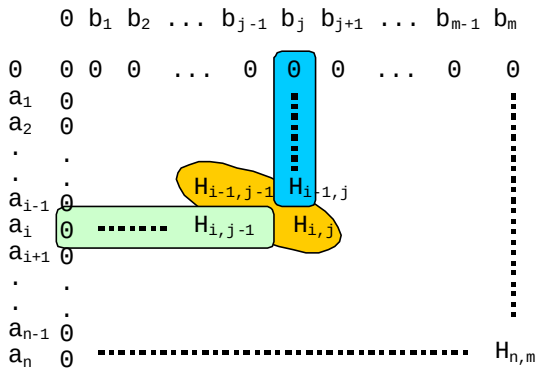


Figure 3 : Matrice du maximum de similarité entre les deux séquences (d'après Smith et Waterman P 173)

La méthode repose sur la construction d'une matrice de score de similarité, où chaque élément correspond au taux de similarité existant entre les deux sous-séquences se terminant par cet élément (Figure 3). Les deux séquences étudiées sont représentées par les ensembles $A = (a_1 , \dots , a_i , \dots , a_n)$ et $B = (b_1 , \dots , b_j , \dots , b_m)$. Un score de similarité s(a,b) est connu entre chaque acide aminé a et b des séquences respectives A et B. Les suppressions d'acide aminé correspondent à un saut vertical ou horizontal dans la matrice. Et W_k est la valeur de pénalisation à appliquer pour compenser cette suppression, où k est la longueur de la suppression, *i.e.* la longueur du saut en nombre d'acides aminés.

Les scores de similarité s (a, b) entre deux acides aminés a et b sont extraits d'une des matrices de similarité disponibles comme les matrices de la série PAM, les matrices BLOSUM ou tout simplement la matrice identité. La fonction de pénalisation W_k d'une suppression de longueur k acides aminés peut être de la forme $W_k = r + kt$, où r est la pénalité appliquée à l'ouverture d'un gap et t la pénalité appliquée à l'extension d'un gap déjà existant.

La matrice obtenue après n x m itérations (Figure 3) est appelée H, et afin de se préserver de valeurs négatives il est postulé que :

$H_{k,0} = H_{0,l} = 0$ avec $0 \leq k \leq n$ et $0 \leq l \leq m$

Chaque élément de la matrice est noté $H_{i,j,}$ et représente la similarité maximale entre les deux segments se terminant l'un par a_i et l'autre par b_j :

$H_{i,j} = \text{Max} (H_{i-1,j-1}+s(a_i , b_j) ; \text{Max} (H_{i-k,j}-W_k) ; \text{Max} (H_{i,j-l}-W_l) ; 0)$

De cette formule il découle que la valeur de la similarité pour a_i et b_j est obtenue de cette manière :

- Si les acides aminés sont similaires alors le cas $H_{i-1,j-1}+s(a_i , b_j)$ s'applique,
- Si l'acide aminé a_i est la terminaison d'une suppression de longueur k acides aminés alors Max $(H_{i-k,j}-W_k)$ s'applique,
- Si l'acide aminé b_j est la terminaison d'une suppression de longueur l acides aminés alors Max $(H_{i,j-l}-W_l)$ s'applique,
- Sinon les acides aminés a_i et b_j n'appartiennent pas à des segments similaires : un zéro est utilisé pour éviter toute valeur négative de $H_{i,j}$.

Ainsi il suffit de rechercher la valeur la plus grande de $H_{i,j}$ pour trouver la terminaison des deux segments des séquences A et B ayant la plus forte similarité. Et pour trouver le début de ces segments , il faut « remonter » la matrice jusqu'à ne plus avoir d'autre possibilité que la valeur zéro. De plus cette analyse fournit automatiquement l'alignement entre ces deux segments. En répétant les même opérations aux régions des séquences A et B autres que les segments similaires déjà identifiés, nous obtenons tous les segments de plus forte similarité des séquences A et B. Il suffit alors de relier ces segments de la meilleure manière possible pour obtenir l'alignement global des deux séquences A et B.

2.3.2 Recherche de protéines similaires : alignement par paire

L'alignement de deux séquences permet d'obtenir le taux de similarité qui les lie. C'est pourquoi la recherche de similarité dans les banques de séquences à partir d'une séquence donnée consiste à réaliser un alignement par paire avec la séquence sujet et chaque séquence de la banque.

L'algorithme de programmation dynamique de Smith et Waterman garantit de trouver l'alignement optimal entre deux séquences. Le logiciel SSEARCH (P 173, P 153) est son implémentation en vue de recherches de similarités dans les banques de séquences. Mais une recherche sur une banque de taille moyenne comme SWISS-PROT demande beaucoup trop de temps pour être envisageable dans le cadre d'un travail de routine. Pour y remédier Shpaer *et al.* (P 171) ont implémenté une version « hardware » de l'algorithme de Smith et Waterman. Réalisation qui permet de conjuguer les garanties que procure cette méthode de recherche de similitude à la rapidité d'exécution inégalable obtenue par une « implémentation hardware ». Mais ce gain est obtenu au détriment de l'universalité puisque l'exécution de la méthode est tributaire d'un matériel spécifique et de plus celui-ci est spécifique de cette unique tâche.

L'autre solution plus universelle est d'ajouter des étapes heuristiques. Une solution est d'éliminer une bonne partie des séquences de la banque en amont de l'utilisation de la méthode de Smith et Waterman. C'est le cas pour FASTA (P 153) et BLAST (P 8) qui sont les deux implémentations heuristiques les plus efficaces et les plus utilisées de cet algorithme. Une autre méthode consiste à appliquer une méthode probalistique à l'algorithme de Smith et Waterman : la méthode PSW (Probabilistic Smith Waterman) (P 33) utilise un système d'attribution de scores aux alignements par paire qui permet d'appliquer une distribution de probabilité sur l'ensemble des paires de séquences.

2.3.2.1 FASTA

FASTA (P 153) permet de classer les séquences d'une banque par ordre de similarité avec une séquence donnée. Pour gagner du temps, la première partie de la méthode est une étape heuristique : elle consiste à rechercher les identités strictes de k-tuples de nucléotides (1-4) ou d'acides aminés (1-2). Ce n'est qu'ensuite qu'intervient une matrice de similarité (matrice de Dayhoff, PAM, BLOSUM, *etc.*) qui permet de calculer un score pour chaque segment de similarité en utilisant l'algorithme de Smith et Waterman.

Le gain en vitesse se fait au détriment de la sensibilité :deux segments similaires comportant un grand nombre de substitutions conservatives ne seront pas repérés. Dans une seconde étape, FASTA tente de concaténer les segments de similarité et aligne les régions inter-segments par une procédure classique. FASTA cherche donc à obtenir un alignement le plus long possible, qui comportera éventuellement des insertions ou des suppressions. Notons que deux segments de

similarité très éloignée (séparés par une longue région sans similarité ou une grande insertion) ne seront pas reliés, un seul segment sera reporté.

2.3.2.2 BLAST

Le programme BLAST (P 8) est plus récent, il cherche également à repérer les segments de similarité entre deux séquences ; mais il diffère de FASTA essentiellement sur deux points : i) les similarités entre acides aminés sont prises en compte dès la première étape, ii) BLAST ne cherche pas à concaténer les segments de similarités. Nous ne considérons que BLASTP qui est l'implémentation de BLAST spécifique à la recherche d'une séquence de protéine sur une banque de séquences de protéine.

Le résultat d'une recherche avec BLAST est donc une liste de segments, non pas un alignement des deux séquences comportant des insertions/suppressions. Tous les segments jugés significatifs seront reportés, quel que soit leur éloignement. Du fait qu'il n'y a pas d'insertions/suppressions, il est possible de calculer *a priori* la probabilité d'obtenir un segment donné et de le conserver ou de le rejeter suivant que son score dépasse ou non un seuil fixé à l'avance. Mais un inconvénient subsiste : si deux protéines se ressemblent « en pointillé » et présentent une succession de petits segments de similarité, ceux-ci risquent d'être rejetés avec cette version de BLAST. Par contre la dernière version de BLAST (BLAST avec gaps) est capable d'introduire des espaces dans la séquence recherchée ou dans la séquence potentielle (P 9). De plus cette version est trois fois plus rapide que la précédente.

PSI-BLAST[10] est une autre implémentation récente de BLAST (P 9). A partir du résultat d'une recherche de similarité de séquences, une matrice recense pour chaque position de la séquence sujet la potentialité d'occurrence de tous les acides aminés. Ensuite une nouvelle étape de recherche est effectuée à l'aide de cette matrice. Et ainsi de suite jusqu'à ce que la recherche converge, c'est-à-dire que ne soient plus trouvées de séquences significativement proches.

2.3.2.3 Efficacité relative des méthodes

La méthode de programmation dynamique de Smith et Waterman (P 173) et la version optimisée de FASTA (P 153) sont plus efficace que BLAST (P 8) pour séparer les faibles similarités du bruit statistique (P 171). Des méthodes plus efficaces utilisent la théorie des modèles cachés de Markov (P 115). Cependant pour des protéines de meilleure similarité, l'efficacité reste comparable.

En considérant le fait que la séquence est complète ou non, *i.e.* si c'est une protéine ou un fragment, des différences notables apparaissent (P 5). Pour les séquences complètes SSEARCH et PSW sont plus efficaces. WU-BLAST2[11] est un bon remplaçant à PSW si le temps de calcul est un facteur limitant. Car le temps de calcul est 20 à 25 fois plus court pour FASTA, et 20 à 35 fois plus court pour BLAST et PSI-BLAST par rapport à l'algorithme de Smith et Waterman (P 152, P 9). Par contre pour les fragments de protéines, BLAST non gappé et WU-BLAST2 obtiennent les meilleurs résultats (P 5).

D'autres alternatives sont par exemple le programme SALSA (P 163). Celui-ci fonctionne quasiment de la même manière que FASTA et BLAST, la différence porte sur la méthode utilisée pour

[10] PSI-BLAST : Position-Specific Iterated-BLAST
[11] WU-BLAST2 : Washington University BLAST2

relier les fragments de plus forte similarité. La méthode implémentée dans SALSA pour relier les fragments est la méthode de programmation dynamique de Smith et Waterman (Chap. 2.3.1). Mais dans ce cas ce ne sont plus les acides aminés qui sont considérés mais les fragments ayant un score de similarité supérieur à un seuil (17). La méthode est alors appliquée entre deux fragments F_i et F_j, candidats potentiels à une liaison. La longueur de chaque fragment est ajustée et leur score modifié en conséquence afin d'obtenir le meilleur alignement possible.

2.3.3 Étude d'une famille de protéines : alignement multiple

Nous avons vu dans le paragraphe précedent comment identifier à l'aide d'alignements par paire des séquences présentant un certain niveau de similarité avec une séquence donnée. L'étape suivante consiste à travailler sur ce groupe de séquences et non plus sur la séquence isolée. Ce travail sur un groupe de séquences proches permet d'intégrer des notions évolutives dans l'analyse de séquence, et de fiabiliser les résultats puisqu'ils concernent alors non plus une séquence mais tout un groupe ayant sans doute des liens de parenté.

C'est pourquoi, le biologiste possède en général un jeu de séquences apparentées qu'il souhaite aligner. C'est-à-dire mettre en concordance les portions adéquates des séquences considérées à l'aide d'une estimation de l'évolution de cette famille de protéines. Se pose alors le problème de l'alignement optimal d'un grand nombre de séquences.

2.3.3.1 Quelques méthodes disponibles : Clustal W, Multalin, *etc.*

Les différents algorithmes d'alignement multiple procèdent toujours plus ou moins de la même manière et sont fondés sur la méthode de programmation dynamique de Smith et Waterman (P 173). Cependant cette méthode ne peut être appliquée pour aligner les N séquences simplement en augmentant le nombre de ces dimensions à N. Sa complexité algorithmique[12] devient rapidement inapplicable avec les matériels informatiques actuels.

2.3.3.1.1 Les algorithmes fondamentaux d'alignement multiple

Certains comme Clustal W (P 180) ou Multalin (P 40) utilisent des algorithmes d'alignement multiple dits progressifs. Ils commencent par réaliser un alignement par paire de chaque séquence avec chacune des autres, c'est-à-dire N(N-1)/2 alignements par paire, puis fondent tous ces alignements un par un pour obtenir l'alignement multiple. Les alignements par paire sont généralement réalisés avec l'algorithme de programmation dynamique (P 173) ou une de ces variantes optimisées (P 139). Les alignements par paire ainsi obtenus permettent de calculer des scores d'identités, et ceux-ci constituent alors une matrice de scores d'identité pour toutes les séquences. À l'aide de cette matrice, un arbre phylogénétique est estimé afin de positionner les séquences les unes par rapport aux autres dans le chemin évolutif que cette famille de protéines a suivi. Et à partir de cet arbre et des alignements par paire réalisés précédemment, un alignement multiple est construit. Cet algorithme est celui employé par exemple pour Clustal W (P 180). Parfois viennent se rajouter de nouvelles étapes à la suite de celles-ci, c'est le cas de l'algorithme Multalin (P 40). Dans l'alignement multiple ainsi obtenu tous les alignements par paires sont de nouveau considérés pour construire une nouvelle matrice de scores. Celle-ci est utilisée pour reconstruire l'alignement multiple. Plusieurs cycles peuvent ainsi être

[12] O(np) pour n séquences de p acides aminés

enchainés afin d'optimiser l'alignement multiple mais généralement il converge après un ou deux cycles (P 40).

L'algorithme MSA (Multiple Sequence Alignment) (P 34, P 130, P 96) met en œuvre un algorithme d'alignement multiple dit simultané. Cet algorithme applique la méthode de programmation dynamique non seulement à chaque paire de séquences mais également à l'ensemble des N séquences à aligner. Ainsi la première étape consiste à calculer le graphe standard de programmation dynamique pour chaque paire de séquences parmi les N. Pour tous les sommets de ces graphes est calculé le coût d'un alignement optimal passant par ces sommets. Ensuite un graphe de programmation dynamique est calculé dans un espace à N dimensions, et MSA ne considère dans ce graphe à N dimensions que les sommets compatibles avec les graphes par paire.

D'autres méthodes comme la méthode SAGA (Sequence Alignment by Genetic Algorithm) (P 143) utilisent un algorithme dit génétique (P 85, P 45). Cet algorithme construit un alignement multiple en mimant les processus évolutifs qui sont supposés avoir conduit la séquence ancestrale à évoluer en les séquences que l'on veut aligner. Le programme DIALIGN (P 137) réalise l'alignement multiple en considérant les segments similaires sans gaps comme le ferait la méthode de la matrice de points.

2.3.3.1.2 Algorithmes optimisant les fondamentaux

Certaines méthodes utilisent ces algorithmes fondamentaux et cherchent à en optimiser l'usage. C'est le cas de DCA[13] (P 176, URL 12) qui applique la méthode de « diviser pour conquérir » à l'algorithme MSA. La méthode DCA sépare chacune des séquences en deux parties. Chaque partie constitue avec ses équivalents de chaque séquence un sous-ensemble. La même division est appliquée à chacun de ces sous-ensembles, et ainsi de suite plusieurs fois si nécessaire. Ensuite dans chaque sous-ensemble, un alignement multiple des portions de séquences est construit. L'alignement multiple des séquences entières est obtenu en raboutant tous les alignements multiples des sous-ensembles.

La méthode COFFEE[14] (P 144) se base sur la méthode SAGA. Elle optimise cette méthode en utilisant une bibliothèque d'alignements par paire de référence et une « fonction objective ». La fonction objective est une fonction de score évaluant la consistence de l'alignement multiple avec les alignements par paire de la bibliothèque de référence. Cette bibliothèque consiste en l'ensemble des N(N-1)/2 alignements par paire possibles avec les N séquences à aligner. Cette bibliothèque est réalisée avec Clustal W comme algorithme d'alignement par paire, car il utilise des pénalités locales d'insertion d'espace (pénalités différentes d'ouverture et d'extension d'insertion d'espaces). La fonction objective utilisée est celle de la méthode SAGA, elle-même dérivée de celle de l'algorithme MSA (Chap. 2.3.3.1.1).

2.3.3.2 Validation de ces méthodes

L'un des problèmes fondamentaux est l'évaluation de la qualité et de la justesse biologique des alignements obtenus avec les différentes méthodes existantes. En effet le biologiste est souvent amené à s'interroger sur la confiance qu'il peut accorder à une méthode suivant les conditions de similarité des séquences considérées. Et un des points cruciaux de cette évaluation est d'avoir une bibliothèque représentative d'alignements multiples optimisés comme référence. Car il a été montré

[13] DCA : Divide and Conquer Alignment
[14] COFFEE : Consistency based Objective Function For alignmEnt Evaluation

que les performances d'un programme d'alignement dépendent du nombre de séquences, du taux de similarité entre les séquences et du nombre d'insertions dans l'alignement (P 131). Récemment, Thompson *et al.* ont construit la bibliothèque BAliBASE (P 181) d'après la connaissance des structures tridimensionnelles des protéines. Cette base d'alignements multiples peut dès lors être utilisée pour tester la qualité des nouveaux algorithmes d'alignement multiple ou pour comparer les anciens. Tous les tests et comparaisons utilisent alors le même référentiel, et non plus des bibliothèques différentes suivant les auteurs.

Briffeuil *et al.* ont étudié la qualité des prédictions de sept serveurs Web (Tableau 4) permettant de réaliser des alignements multiples de séquences de protéine (P 31). Ils ont comparé ces méthodes sur des critères de puissance (sensibilité) et de confiance (sélectivité). Les sept méthodes évaluées sont Clustal W (P 180), MAP (P 107), PIMA (P 172), Block Maker (P 99) ,MSA (P 130), MEME (P 91) et Match-Box (P 56, P 55) (*cf.* Tableau 4 pour les adressses Web). Ces méthodes ont été choisies parce qu'elles représentent les deux catégories d'alignement multiple : globale (Clustal W, MAP, PIMA, MSA) et locale (MEME, Block Maker, Match-Box). Cette évaluation a utilisé les paramètres par défaut pour chaque méthode et une bibliothèque d'alignements multiples non-gappés de référence. Cette base de référence est constituée des alignements multiples améliorés manuellement de 20 familles de protéines ayant de faibles niveaux d'identité. Chaque famille a au moins trois structures connues, la même fonction, contient un cœur commun représentatif de portion de séquence et inclut au moins deux séquences de faible identité.

Tableau 4 : URL des serveurs d'alignement multiple comparés en utilisant les paramètres par défaut, d'après Briffeuil *et al.* (P 31).

Méthodes	**URLs**
Block maker	http://www.blocks.fhcrc.org/blockmkr/make_blocks.html
Clustal W 1.6	http://dot.IMGEN.bcm.tmc.edu:9331/multi-align/options/clustalw.html
Match-Box	http://www.fundp.ac.be/sciences/biologie/bms/matchbox_submit.html
MAP	http://dot.IMGEN.bcm.tmc.edu:9331/multi-align/options/map.html
MEME	http://www.sdsc.edu/MEME/meme.1.4/meme.nofeedback.html
MSA	http://alfredo.wustl.edu/msa.html
PIMA	http://dot.IMGEN.bcm.tmc.edu:9331/multi-align/options/pima.html

Il ressort de cette évaluation que, pour les conditions utilisées, Clustal W, MAP, MSA et PIMA sont deux fois plus puissants que Block Maker et MEME (80-90% contre 25-45%), alors que leur taux de confiance sont très semblables (65-75%). Match-Box est différent puisqu'il propose une valeur de solidité de l'alignement comme paramètre de calcul. Ainsi suivant la valeur de ce paramètre, ces résultats sont situés entre ces deux positions : soit entre les deux groupes précédents (puissance 70%, confiance 78%), soit à une valeur de confiance très élevée (95%) pour une puissance plutôt faible (25%) (Figure 4).

Puissance = L/S

Confiance = L/s

L : longueur cumulée des segments correctement prédits.

S : longueur cumulée des segments structuraux conservés.

s : longueur cumulée des segments structuraux conservés sans espaces (gaps).

Cependant un défaut des méthodes Clustal W, MAP, MSA et PIMA est qu'elles ne proposent pas dans leur fichier de sortie des données permettant une analyse de la solidité de l'alignement obtenu. Il faut utiliser d'autres logiciels pour juger de la valeur de l'alignement.

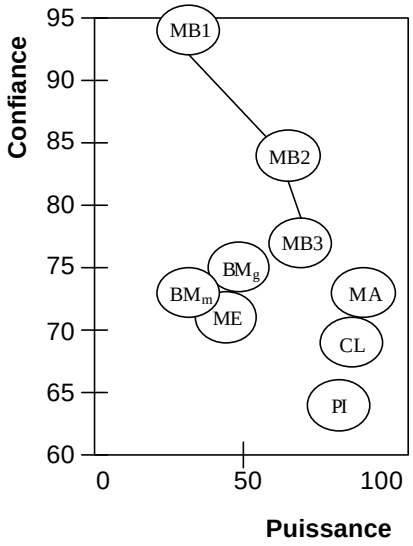


Figure 4 : Comparaison des différents serveurs d'alignements multiples, d'après Briffeuil *et al.* (P 31)

MSA à la différence des autres méthodes est une méthode exacte alors que les autres sont des méthodes heuristiques d'alignement multiple. La principale conséquence est une quantité de calculs à réaliser beaucoup plus grande, d'où découle une limitation du serveur de MSA à moins de 800 séquences ou à une durée maximale de calcul de 10 minutes. Alors que la qualité des alignements obtenus est équivalente à celle des alignements construits par Clustal W, qui apparaît comme une de méthodes ayant les meilleures potentialités.

2.4 Prédiction de structure secondaire

La structure d'une protéine donne beaucoup d'informations sur sa fonction et sa régulation. Mais ni la cristallographie aux rayons X ni la RMN n'est la panacée. En effet la première technique nécessite que la protéine étudiée cristallise, ce qui peut prendre plusieurs mois ou même ne jamais se réaliser. La seconde technique, la RMN, est limitée à des fragments d'une centaine d'acides aminés ; et l'échantillon doit être soluble à forte concentration. Ainsi il est plus difficile, voire impossible pour l'instant, d'étudier des structures plus grosses, comportant des sous-unités ou flexibles que des plus petites, globulaires et rigides. De plus ces techniques demandent de longues et coûteuses manipulations, ce qui n'est pas conciliable avec le nombre de protéines dont la connaissance de la structure présente un intérêt certain. C'est pourquoi, la détermination de structure de protéines par des outils bioinformatiques s'est rapidement révélée une nécessité ; et des techniques comme la modélisation moléculaire par homologie ou la prédiction de structure tridimensionnelle *de novo* se développent.

En effet, le repliement d'une protéine est déterminé par sa séquence d'acides aminés (P 10), et par des facteurs liés à l'environnement moléculaire dans lequel elle se replie comme les molécules chaperonnes (P 62, P 98, P 141). Cependant l'information apportée par la séquence d'acides aminés a une importance relative plus grande, et fort de ce constat des méthodes essayent depuis plus de 20 ans de prédire la structure tridimensionnelle des protéines. Ce but est loin d'être atteint pour la structure complète d'une protéine ; cependant les taux de confiance sont relativement bons pour la prédiction de structure secondaire, de l'ordre de 70-75% (P 165, P 51) pour trois états conformationnels.

La prédiction de structure secondaire de protéines peut être un bon point de départ lors d'une prédiction de repliement d'une protéine. Elle consiste alors en une étude des arrangements possibles de

ces structures secondaires. La prédiction de structure secondaire est aussi utile dans le cadre d'une modélisation par homologie puisqu'elle permet de vérifier si les structures secondaires de la protéine étudiée sont les mêmes et ont le même arrangement spatial que celles d'une protéine de structure connue, *i.e.* extraite de la PDB (P 1).

Mais les structures secondaires sont aussi utiles lors de l'étude de l'évolution d'une séquence, c'est-à-dire à l'étude d'une famille de protéines. En effet il est connu que les structures évoluent moins vite que les séquences grâce à des variations compensatoires et une certaine plasticité des structures secondaires (P 124), et que la structure secondaire des protéines est mieux conservée que la structure atomique précise (P 135). De plus le processus de substitution des acides aminés au cours de l'évolution est significativement corrélé à l'environnement en structure secondaire (P 87).

2.4.1 *Prédiction à partir de la seule séquence de la protéine*

Les plus anciennes recherches relatives à la prédiction de la structure secondaire des protéines datent des années 60. Ces premières méthodes développées étaient des méthodes statistiques et ce même si à l'époque la taille de l'échantillon des protéines de structure connue était très faible. Elles concernaient alors essentiellement l'identification de résidus permettant la localisation d'hélices (les "helix formers" et "helix breakers").

Cependant, les premiers résultats significatifs ont été obtenus dans les années 70, avec en particulier les travaux de Chou et Fasman (P 38, P 39, P 65). Ils ont calculé la propension (rapport de fréquences) de rencontrer chaque acide aminé dans un état structural donné (hélice, brin β, coude et région apériodique). Plusieurs améliorations de la méthode ont été par la suite proposées : distinguer les brins β internes et externes (P 75), utiliser la prédiction de la classe structurale (P 53) et enfin utiliser un profil dans le cas particulier des protéines β–α–β (P 179).

D'autres auteurs ont appliqué la théorie de l'information directionnelle (P 74), puis ont considéré les paires d'acides aminés le long de la séquence et non plus les acides aminés pris un à un (P 83) : c'est la méthode GOR. Ses paramètres ont été actualisés pour la version la plus récente : GOR 4 (P 73).

2.4.2 Incorporation de l'information relative à l'évolution de la séquence

2.4.2.1 En utilisant la notion de similarité de séquences

A la même époque, les méthodes basées sur la similarité de séquences sont apparues (P 126, P 177). Brièvement, chaque polypeptide de la séquence échantillon est comparé avec tous les peptides de même longueur de la base de référence. Si deux peptides sont dits similaires (en utilisant une matrice de substitution de structures secondaires) la structure secondaire expérimentale est attribuée au peptide considéré avec le score de similarité. Le processus est poursuivi jusqu'à ce que tous les peptides aient été comparés, les scores étant alors additionnés. L'état structural final prédit est celui présentant le score conformationnel le plus élevé. Cette stratégie a été utilisée dans de nombreuses méthodes (P 126, P 125) et combinée avec une procédure d'auto-optimisation (P 79). Plus récemment, plusieurs groupes ont utilisé une approche basée sur les réseaux de neurones (P 157, P 167, P 188).

2.4.2.2 En utilisant les alignements multiples

D'autres méthodes tiennent compte de l'information contenue dans un alignement multiple de protéines homologues afin d'augmenter de manière significative la qualité de la prédiction. Cette approche a été mise en œuvre en utilisant les différentes méthodes déjà développées : (i) les méthodes statistiques (P 58), (ii) les méthodes basées sur la similarité, *e.g.* SIMPA (P 127, P 60), (iii) les méthodes dites d'auto-optimisation, *e.g.* SOPMA (P 81) et enfin (iv) les méthodes basées sur les réseaux de neurones, *e.g.* PHD (P 167, P 164).

2.4.2.3 En utilisant la Phylogénie

Puisque la structure tridimensionnelle des protéines évolue moins vite que leur séquence (P 124, P 135) et le processus de substitution des acides aminés au cours de l'évolution est significativement corrélé aux structures secondaires présentes (P 87), certaines méthodes se fondent sur l'utilisation d'arbres phylogénétiques pour la prédiction de structure secondaire et d'autres analyses comparatives de séquences (P 86). D'autres méthodes utilisent des modèles cachés de Markov (HMM) pour une combinaison de la phylogénie et de notions structurales afin de prédire l'arbre phylogénétique, l'alignement des séquences et leur structure secondaire à partir d'une structure tridimensionnelle connue (P 129).

2.4.3 *Comparaison des qualités de prédiction*

La comparaison objective de la qualité des méthodes de prédiction nécessite l'utilisation d'une même base de test constituée selon des critères identiques. Un des critères important est le taux d'identité maximal entre chacune des séquences constitutives. Il est aujourd'hui admis que ce seuil doit être de 25% d'identité dans un alignement par paire pour des séquences de plus de 80 résidus (P 169). Une valeur en dessous de laquelle la modélisation moléculaire par homologie ne peut plus être appliquée sans risque. Dans ces conditions la méthode PHD (P 166, P 167, P 164) a une qualité de prédiction selon 3 états conformationnels (Q_3) de 71.4%, les méthodes GOR IV (P 73), SIMPA (P 127) et SOPMA (P 81) ont respectivement un Q_3 de 66.5%, 69.4% et 69.7% (après lissage). Ces dernières qualités sont obtenues sur une base de test comportant 609 protéines, alors que la qualité obtenue par PHD a été obtenue sur une base plus ancienne ne contenant que 126 chaînes protéiques. Il convient de souligner que si les méthodes GOR 4, SIMPA et SOPMA sont combinées, le Q_3 est alors de 71.3% sur la base de protéines.

2.4.3.1 Nouvelles perspectives

Il semble cependant clair aujourd'hui, que les prédictions de structure secondaire de protéines arrivent à un palier (P 165), et qu'il ne pourra y avoir de gains significatifs de la qualité de prédiction sans tenir compte d'autres facteurs comme par exemple les effets du repliement global, *i.e.* la structure tridimensionnelle, sur le repliement local, la structure secondaire. Cependant une amélioration peut être obtenue en combinant plusieurs méthodes (P 26, P 54, P 166). C'est le cas de la méthode MLR (P 94) et de celle mise en œuvre sur le serveur JPred (P 43). Elles permettent respectivement d'obtenir une qualité de prédiction Q_3 de 71,3% et de 72,9%.

Comme nous l'avons évoqué précédemment, les méthodes actuelles de prédiction de structure secondaire ont au maximum un taux de confiance d'environ 70% pour trois états conformationnels. Chaque acide aminé de la séquence a donc 70% de chances d'être prédit dans le bon état conformationnel (P 81, P 57, P 164, P 168, P 51). Il est donc indispensable de tenir compte des 30%

d'imperfection des méthodes de prédiction de structure secondaire de protéine lors de la mise en œuvre d'algorithmes ou de méthodes d'analyse biologique utilisant ces prédictions de structure secondaire (P 36), par exemple lors de modélisations moléculaires.

2.5 Outils bioinformatiques existants

Nous allons maintenant considérer les logiciels existants dont le domaine d'application se rapporte à l'analyse de séquences de protéine. Pour ce faire , nous utilisons le catalogue des logiciels et serveurs biologiques « BioCatalog[15] » (P 161, P 162). Un site Web lui est consacré (URL 53) à l'EBI (URL 14).

BioCat répertorie les logiciels et serveurs utiles à la biologie moléculaire et à la génétique. La version actuelle (version 6.0 du 2 mars 1999) répertorie 580 logiciels et serveurs. BioCat est mis à jour de deux manières : soit par une inscription directe d'un nouveau logiciel par les auteurs, soit par une veille technologique. Cette veille porte sur les différents média que sont les listes de nouvelles électroniques, les serveurs ftp, les publications papiers ou électroniques. Elle est réalisée par les personnes qui maintiennent BioCat et par des experts internationaux de la Bioinformatique tels Dominique Caterina, Amos Bairoch ou Philippe Dessen.

Les logiciels répertoriés dans BioCat sont répartis en différents domaines qui caractérisent les différentes applications possibles de la Bioinformatique. Ainsi 11 domaines et 53 sous-domaines sont définis (P 161). Mais nous ne nous intéresserons qu'aux 12 sous-domaines (Tableau 5) qui ont plus particulièrement trait à l'analyse de séquences de protéine. La liste exhaustive des logiciels (au nombre de 219) que nous considérons est indiquée dans l'Annexe A, ainsi que les références bibliographiques correspondantes.

Tableau 5 : Les domaines de BioCat et leur nombre d'entrées

Domaine	**Nombre d'entrées**	**Domaine**	**Nombre d'entrées**
Alignment browser	12	Pattern identification	31
Alignment editing and display	28	Protein sequence analysis	46
Alignment search software	105	Searching databases	44
Comparative analysis	7	Sequence analysis	93
Database and analysis	25	Sequence format conversion tools	25
GCG tools	25	Structure prediction	21

Parmi les 219 logiciels considérés, 180 ont 1 ou 2 domaines d'application biologique du fait des différentes fonctionnalités qu'ils proposent (Tableau 6) et seulement 39 logiciels ont plus de 2 domaines d'utilisation. De plus 135 (61,6%) de ces logiciels ne fonctionnent que sur un seul type de système d'exploitation, avec une grosse majorité pour UNIX (Tableau 7). Les logiciels supportant différentes architectures logicielles sont en faible nombre : 27 (12,3%). De ces chiffres se détache une nette prépondérance pour les logiciels fonctionnant sous UNIX, donc sur des stations de travail. Ce matériel est très peu répandu dans les laboratoires de biologie. Par contre les ordinateurs individuels ne disposent que d'un plus faible nombre de logiciels d'analyse de séquences de protéine : 43 logiciels

[15] initié en 1992 par Dominique Caterina du CPEH-Fondation Jean Dausset en collaboration avec le Généthon

(19,6%). Cependant ces chiffres sont à pondérer par le fait que pour 25% des logiciels considérés (55 logiciels) l'information du système d'exploitation supporté n'est pas disponible.

Tableau 6 : Biocat, répartition des 219 logiciels cons idérés suivant leur nombre de domaines d'utilisation

Nombre de domaines d'utilisation d'un logiciel	1	2	3-4	5 et +
Nombre de logiciels concernés	129	51	20	19

Tableau 7 : Biocat, répartition des 219 logiciels considérés suivant les systèmes d'exploitation supportés

Environnement	**Occurrences**	**Taux** (sur 219 logiciels)
PC seul	9	4,1%
Macintosh seul	7	3,2%
UNIX seul	119	54,3%
PC et Mac	2	0,9%
UNIX et PC	12	5,5%
UNIX et Mac	7	3,2%
UNIX, PC et Mac	8	3,6%
Non-accessible	55	25,1%

Le langage utilisé pour écrire ces logiciels est en nette majorité le langage C : 114 logiciels (52%). Ensuite vient le Fortran (14%), puis en proportion moindre le Perl et le Pascal (5% chaque). Une proportion non négligeable (13,2%) de ces logiciels utilisent plusieurs langages. Ceci traduit l'évolution des logiciels au cours du temps, mais également l'utilisation de bibliothèques écrites par d'autres programmeurs et mise à la disposition de la communauté de bioinformaticiens. Ici encore, pour une bonne partie des logiciels (64 logiciels soit près de 30%), l'information n'est pas disponible ou erronée[16].

Langage utilisé	**Occurrences**	**Taux** (sur 219 logiciels)
Langage C	114	52%
Fortran	30	13,7%
Perl	12	5,5%
Pascal	10	4,6%
Non-accessible	64	29,2%
Divers	18 (VB 2, Lisp 2, yac 1, lex 1, TclTk 2, Csh 4, awk 1, C++ 2, Javascript 2, Java 1)	8,2%

[16] Il y a 3 occurrences du langage « English »

3 Matériels et méthodes

Nous allons maintenant évoquer les différents matériels et logiciels utilisés pour nos développements d'outils et méthodes bioinformatiques et pour les services que nous proposons à la communauté scientifique à travers le Web.

3.1 Machines

Au laboratoire nous utilisons plusieurs types de machines différentes (Tableau 8). Ces matériels servent à la fois aux développements et à la mise en œuvre des services. Ces machines ont des architectures et des interfaces logicielles très diverses puisque notre parc informatique est composé d'ordinateurs personnels mono-processeur (Mac PowerPC 8200, PC G6-450) comme de stations de travail quadriprocesseurs avec mémoire unifiée (Silicon Graphics Origin 200 et Origin 2000).

Tableau 8 : Matériels informatiques utilisés pour les développements et les services au laboratoire

Nom	**Série**	**Processeur Type (nombre)**	**Fréq. (MHz)**	**RAM (Mo)**	**Disque dur (Go)**	**Système d'exploitation**	**Marque**
Triumph	Indigo	R 4400 (1)	180	64	2	IRIX 5.3	SGI
Ferrari	Origin 2000	R10 000 (4)	195	1 000	54	IRIX 6.4	SGI
Bmw	Origin 200	R10 000 (4)	225	512	118	IRIX 6.5	SGI
Bugatti	Risc 6000	PowerPC 604 (1)	195	64	2	AIX 4.1	IBM
Pc-mod	G6-450	P II 450 (1)	450	128	13.6	Windows 98	Gateway 2000
Mac-cge	PowerPC 8200	PowerPC 601c (1)	100	96	4	MacOS 8	Macintosh

3.1.1 « Software » : Interface logicielle

Une grosse différence entre les stations de travail et les micro-ordinateurs est l'architecture multitâche et multiutilisateur des premières alors que les seconds sont mono-utilisateurs[17] et depuis peu seulement multitâches[18]. Mais cette différence est essentiellement due au système d'exploitation employé plus qu'à l'architecture matérielle. En effet il est maintenant possible d'installer sur n'importe quel ordinateur individuel[19] un ensemble logiciel adéquat qui lui apporte toutes les fonctionnalités logicielles d'une station de travail.

Les micro-ordinateurs Macintosh® ou PC utilisent par défaut un système d'exploitation propriétaire dont l'évolution est limitée aux développements de son éditeur. Ces systèmes sont principalement MacOS pour les Macintosh et Windows 3.1/95/98 pour les micro-ordinateurs compatibles PC. Il est à noter que l'assemblage des composants et la conception du système d'exploitation des Mac sont assurés par la même entreprise Apple Systems Inc. ; alors que pour les PC les assembleurs sont nombreux (IBM, HP, Gateway, Dell,…) bien que l'éditeur du système d'exploitation soit unique Microsoft Corp. Ces systèmes d'exploitation intègrent une interface graphique fenêtrée qui gère les relations entre les processus. Et il n'y a pas d'accès préemptif sur ces processus.

[17] Bien que certains systèmes d'exploitation récents comme Windows NT® et Linux confèrent aux ordinateurs individuels les fonctions multi-utilisateurs des stations de travail.

[18] La possibilité d'être multitâches est apparue avec MacOS 8 et Windows 95.

[19] Sous réserve que les pilotes (drivers) existent.

Les stations de travail utilisent un système d'exploitation qui n'est pas propriétaire à la base, mais développé par toute la communauté informatique au travers de diverses associations comme le projet GNU (URL 21) de la Free Software Foundation (FSF) à l'Université de Cambridge, Massachussett. Cependant UNIX n'est pas unique et l'on retrouve donc des différences dans les UNIX livrés par les constructeurs informatiques et dans ceux proposés gratuitement sur Internet. Ces différences peuvent concerner les commandes, parfois essentielles (par exemple *tar*[20]), ou les outils disponibles tel le debugger *dbx* (*cf.* 3.3.5). Les UNIX que nous utilisons pour nos développements et services sont IRIX™ 5.3, IRIX™ 6.4, IRIX™ 6.5 et AIX™ 4.1. Nous utilisons également le système d'exploitation Solaris 5.6 au travers d'un compte de connexion sur la machine 'lovelace.infobiogen.fr ' du serveur national GIS-InfoBioGen.

Une ouverture importante du monde UNIX est actuellement en train de se mettre en place en direction du monde des micro-ordinateurs. Elle est due principalement au système d'exploitation Linux initié par Linus Torvalds en 1991[21] mais également à d'autres systèmes d'exploitation de type UNIX comme FreeBSD, NetBSD ou OpenBSD. Il est maintenant possible d'installer sur n'importe quel micro-ordinateur un système d'exploitation multitâche et multiutilisateur de type UNIX. Mais ce n'est parfois pas aisé et nécessite des modifications ou paramétrages en profondeur notamment lorsque l'architecture matérielle est récente et les pilotes n'existent pas.

3.1.2 « Hardware » : Architecture matérielle

Les premiers matériels que nous évoquerons sont les architectures monoprocesseurs utilisées pour le développement et portage de la bibliothèque « biolcp » et du logiciel MPSA. Il s'agit d'un Macintosh PowerPC 8200, d'un Gateway G6-450, d'une station de travail SGI Indigo et d'une station IBM Risc 6000 (Tableau 8). Toutes ces machines possèdent un seul processeur cadencé à différentes vitesses (de 100 à 450 MHz), relié à une ou plusieurs unités de mémoire (2 « barettes mémoire » dans le cas du Macintosh PowerPC) et à divers périphériques.

Les autres machines que nous utilisons sont des serveurs multiprocesseurs sous UNIX. Ces deux machines sont des Silicon Graphics Origin 200 et Origin 2000 (Tableau 8). Elles sont équipées chacune de 4 processeurs. Les processeurs sont montés par paire sur une carte mère biprocesseur appelée « nœud élémentaire ». Les nœuds sont ensuite reliés entre eux par un bus de type CrayLink Interconnect™ à 800 Mbit/s. Les unités de mémoires sont réparties sur les nœuds mais partagées par l'ensemble. Cette architecture est dite S2MP (Scalable Shared-memory MultiProcessing). Ainsi les deux machines que nous possédons (de nom d'hôte « bmw » et « ferrari ») ont chacune 4 processeurs et une seule mémoire disponible pour les 4. Cette mémoire se répartit automatiquement suivant les besoins ponctuels manifestés par chacun des quatre processeurs. Elle est de 1 Go pour ferrari et de 512 Mo pour bmw.

3.1.3 Avantages d'un parc hétérogène pour le développement logiciel

Cette diversité des architectures matérielles et logicielles au sein de notre laboratoire se veut comme un reflet de l'hétérogénéité du parc informatique des laboratoires de Biologie. Elle nous

[20] La commande *tar* a sous AIX et Solaris l'option *-h* qui force le suivi des liens symboliques, alors que sous IRIX l'option *-h* force au contraire à ne pas suivre les liens symboliques (l'option *-L* force ce suivi sous IRIX).
[21] et repris depuis par toute la communauté informatique.

permet de tester la portabilité des applications que nous développons et que nous proposons à la communauté scientifique.

De plus cette hétérogénéité est implicite de par les différents tâches que nous effectuons quotidiennement. En effet les machines monoprocesseurs sont plutôt utilisées pour le développement ou comme poste de travail utilisateur lors d'une analyse de séquences. L'indigo (nom d'hôte « triumph », Tableau 8) sert au développement initial des programmes en langage C. Leur portabilité est ensuite testée sur les autres machines dans les différents systèmes d'exploitation (*cf.* 3.1.1). Par contre les deux machines quadriprocesseurs (« bmw » et « ferrari »; Tableau 8) sont dévolues pour tout ou partie au serveur Web NPS@ que nous avons mis en place au sein du PBIL (URL 35) ainsi qu'au serveur de requêtes MPSAweb.

3.2 Réseaux

Un réseau informatique existe lorsqu'au moins deux objets informatiques sont reliés. Cinq catégories de réseaux peuvent être dénombrées d'après le critère de la longueur de la liaison (Figure 5). Et plusieurs modèles sont définis pour organiser l'échange des données entre les différents points d'un réseau : OSI, TCP/IP, ATM, …

Nous utilisons le modèle TCP/IP à travers la bibliothèque *socket* pour établir une connexion entre le client MPSA et le serveur d'analyse. Et les données sont échangées entre le serveur et les clients à l'aide du Web (HTTP, HTML, types MIME).

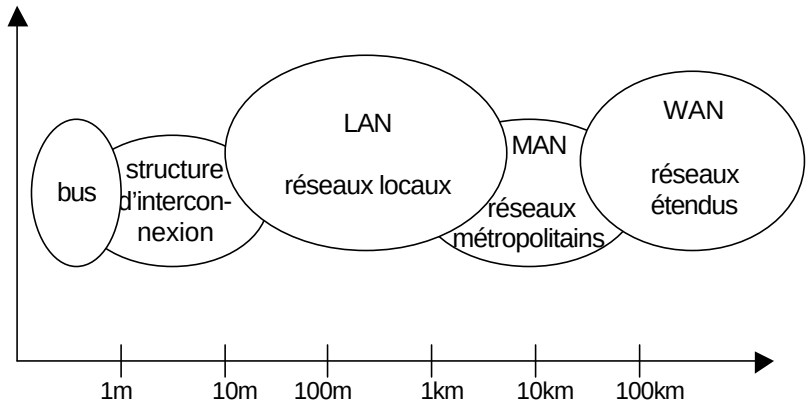


Figure 5 : Les réseaux informatiques

3.2.1 Les différentes catégories de réseaux informatiques

Les réseaux sont répartis dans différentes catégories en fonction de leur longueur (Figure 5). Les deux premières catégories sont majoritairement des liaisons parallèles alors que les trois autres sont des liaisons séries.

Les bus relient les processeurs, les unités de mémoire, les entrées-sorties d'un calculateur ou d'un multiprocesseur. Les structures d'interconnexion relient à des distances faibles et avec un très haut débit différents calculateurs. Les réseaux locaux que l'on appelle aussi LAN correspondent par leur taille aux réseaux de campus. La distance classique pour couvrir les bâtiments à câbler est de plusieurs centaines de mètres. Les débits sont aujourd'hui de quelques Mbit/s à quelques dizaines de Mbit/s.

Les réseaux métropolitains ou MAN correspondent à une interconnexion de plusieurs bâtiments situés dans une même ville. Ils doivent être capables d'interconnecter les LAN des différents bâtiments et de prendre en charge les machines communes à l'ensemble de la gestion du site distribué. Enfin, les réseaux étendus ou WAN sont destinés à transporter des données numériques sur des distances à l'échelle d'un pays. Tous ces réseaux étendus sont à leur tour interconnectés pour constituer le réseau des réseaux : Internet.

Les liaisons de ces différents réseaux sont soient physiques (utilisant des signaux électriques ou optiques) soient hertziennes[22]. La technique de transfert qui a été choisie pour Internet s'appelle la commutation de paquets : toutes les informations sont découpées en fragments qui sont transportés à l'autre extrémité du réseau. Les paquets sont de tailles variables avec des tailles maximales d'unité de transfert (MTU[23]), variables suivant la couche physique du réseau (Éthernet, Token-Ring, FDDI[24],…) ou fixes[25] comme dans le cas de réseau ATM. Afin de normaliser tous les composants nécessaires à la marche d'un réseau à commutation de paquets, l'organisation internationale de standardisation (ISO[26]) a proposé une décomposition de l'architecture d'un réseau en 7 niveaux : le modèle de référence OSI.

3.2.1.1 Le modèle de référence OSI

Le modèle OSI décompose l'architecture d'un réseau en 7 couches (Figure 6). Chaque couche N doit être capable de communiquer avec la couche N-1 et la couche N+1. Un protocole de niveau N définit un ensemble de règles nécessaires pour permettre le transport des informations d'un niveau N à un autre niveau N. De plus les règles utilisées pour contrôler l'envoi des données sont variables suivant le niveau N du protocole.

Ainsi lorsqu'une application de la machine A veut échanger des données avec la machine B (Figure 6), celles-ci descendent de la couche application (niveau 7) vers la couche physique (niveau 1), transitent par le réseau et remontent de la couche physique à l'application de la machine B. Ce modèle garantie un processus normalisé quel que soit le support d'interconnexion (la couche physique) du réseau concerné (Ethernet, FDDI, Token-Ring,…). Mais il est en fait peu utilisé sur Internet où prime quasi majoritairement un autre modèle : le modèle TCP/IP.

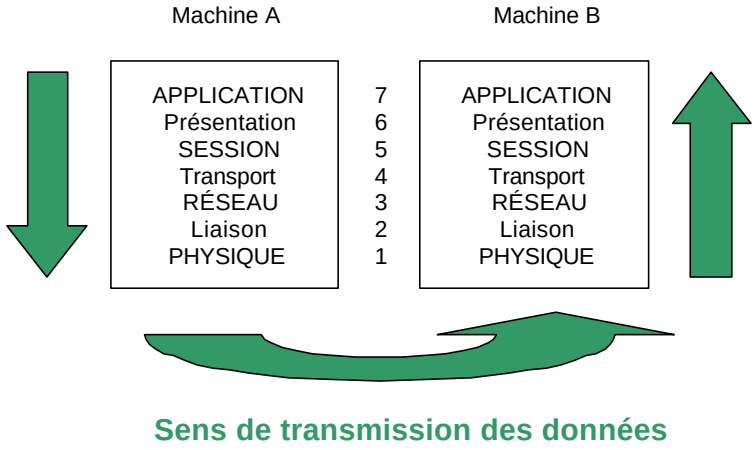


Figure 6 : Modèle OSI : principe de l'échange de données entre deux machines

[22] pour les MAN et les LAN
[23] MTU : Maximal Transfert Unit
[24] FDDI : Fiber Distributed Data Interface
[25] on parle alors non plus de commutation de paquets mais de commutation de cellules
[26] ISO : International Standardization Organization

3.2.1.2 Le modèle TCP/IP : fondement d'Internet

Le modèle TCP/IP est le modèle quasi universel d'architecture réseau de l'Internet. TCP/IP est un couple de protocoles : « Internet Protocol » (IP) est un protocole de niveau 3 et « Transfert Control Protocol » (TCP) un protocole de niveau 4 dans le modèle OSI. Mais le modèle TCP/IP regroupe également plusieurs autres protocoles (Figure 7).

Un autre protocole de niveau transport disponible est UDP, mais nous ne l'utiliserons pas car UDP est un protocole non-connecté. Il est utilisé pour des applications où la rapidité de l'échange des données prime sur sa fiabilité.

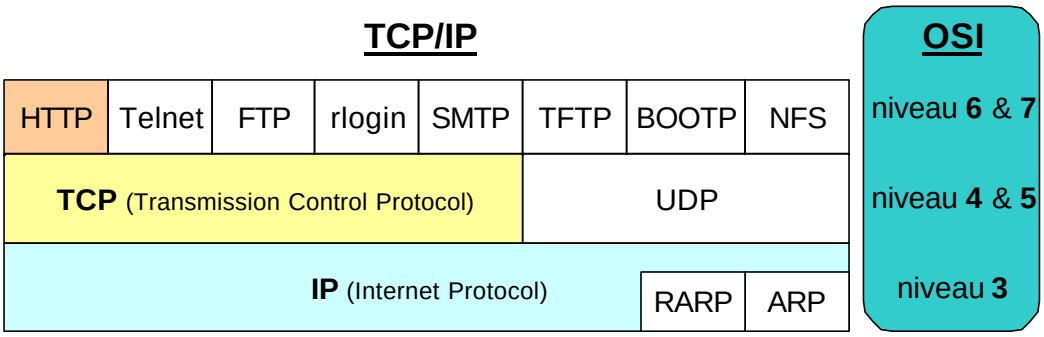


Figure 7 : Architecture TCP/IP et correspondance avec le modèle OSI

3.2.1.2.1 Internet Protocol

IP a pour but de transporter les paquets, appelés datagrammes, d'une extrémité à l'autre du réseau. IP est un protocole non-connecté : les datagrammes sont envoyés sans établir de connexion avec la machine destinatrice et sans se soucier de leur arrivée sans erreur. Les paquets sont indépendants les uns des autres et sont routés individuellement dans le réseau : le chemin suivi n'est pas connu à l'avance et dépend des routeurs rencontrés pour chaque paquet[27].

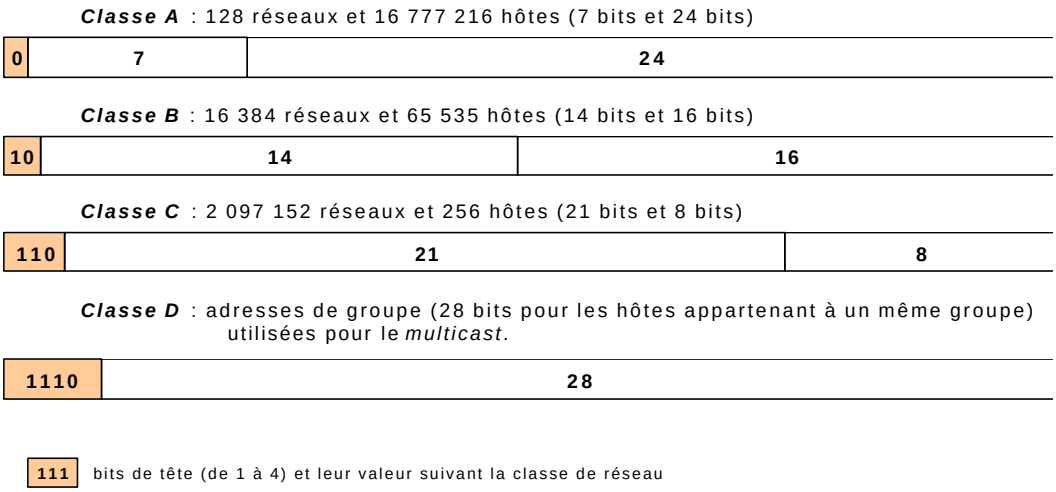


Figure 8 : Les zones d'adresse IP

[27] On dit qu'Internet est un réseau routé par comparaison avec les réseau X.25 et ATM qui sont dits commutés où le chemin suivi par les paquets y est toujours le même.

Pour identifier chaque machine connectée à Internet, IP lui attribue une adresse codée sur 4 octets. Cette adresse IP est différente de l'adresse physique, ou adresse MAC[28], fixée en usine lors de la fabrication de la machine. La correspondance entre ces deux types d'adresse est effectuée par deux protocoles intégrés à IP : ARP et RARP[29]. Les 32 bits d'une adresse IP sont répartis entre une adresse de réseau et une adresse d'hôte[30]. Cette répartition variable des bits définit différentes classes de sous-réseaux IP en fonction du nombre d'hôtes qu'ils peuvent héberger (Figure 8). Les adresses de classe D sont un peu particulières : elles sont utilisées pour définir les groupes de *multicast*.

Les quatre octets d'une adresse IP sont séparés par un point lors d'une représentation graphique : par exemple 193.51.160.1. Cette forme d'adresse IP n'est pas très facile à retenir ni très simple à manipuler. Une autre forme de l'adresse est parfois disponible : cette forme est appelée adresse littérale ou mnémonique. Elle contient le nom de l'hôte et le nom de son domaine[31] libellés généralement avec des mots : par exemple l'adresse littérale de la machine 193.51.160.1 est 'bentley.ibcp.fr'. La correspondance entre les deux formes d'une adresse IP est effectuée par le service de nom (DNS).

IP est la couche réseau du modèle TCP/IP. Au dessus intervient le protocole TCP qui se charge de la couche transport et session.

3.2.1.2.2 Transfert Control Protocol

TCP est un protocole fiable par rapport à IP. Le protocole TCP voit les données comme un flot d'octets divisés en segments. Ces segments sont échangés suivant un circuit virtuel en mode connecté : TCP établit une connexion bidirectionnelle avec la machine cible, contrôle le flux de données et ré-expédie les données qui ne sont pas parvenues à destination.

TCP permet également à plusieurs programmes d'établir une connexion simultanément et démultiplexe les données reçues d'applications différentes. TCP utilise pour ceci la notion abstraite de port qui identifie la destination ultime dans la machine cible.

TCP est principalement un protocole réseau d'après le modèle OSI. Mais il dispose également de fonctions relatives à la couche session. TCP assure donc les services de niveau 4 et 5 du modèle OSI. Les services des niveaux présentation et application sont assurés par différents protocoles ou applications dans le modèle TCP/IP (Figure 7) : HTTP, SMTP, FTP,… Chacun de ces services utilise un port TCP qui lui est propre : par exemple HTTP utilise le port 80 et SMTP le port 25. Nous nous attarderons sur HTTP que nous utilisons pour la mise en place d'outils client-serveur.

Du fait de l'existence de ces différents protocoles au dessus de TCP , une extension des adresses IP littérales a été effectuée : il s'agit de l'URL (Uniform Ressource Locator). Un URL permet de définir précisément une connexion Internet[32] en spécifiant le protocole utilisé, l'adresse IP de la machine cible et le fichier ou système de fichiers visés (*e.g.* http://www.ibcp.fr/mpsa).

[28] MAC : Media Access Control
[29] ARP : Address Resolution Protocol ; RARP : Reverse Address Resolution Protocol
[30] Sauf pour les réseaux de classe D.
[31] Le domaine est parfois divisé en sous-domaines.
[32] Suivant le modèle TCP/IP ;

3.2.1.2.3 HyperText Transfert Protocol

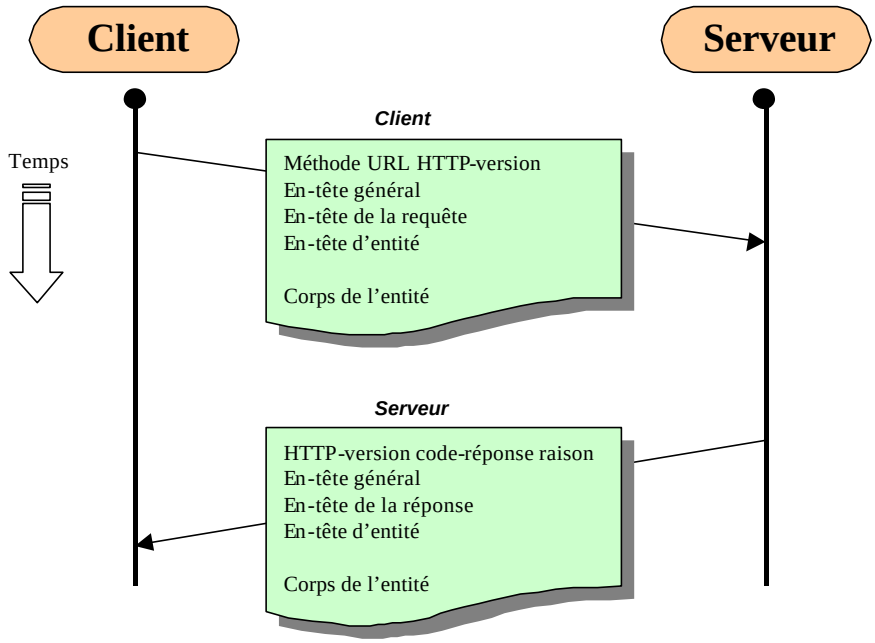


Figure 9 : Format des transactions HTTP.

HTTP (Hyper-Text Transfert Protocol) est le protocole développé au début des années quatre-vingt-dix pour servir de support au Web. Ce protocole utilise TCP/IP pour la mise en forme des données et la gestion de leur transit. HTTP est défini par les spécifications[33] diffusées par le W3C (URL 56). La version actuelle de HTTP est la version 1.1 définie par la RFC 2068.

HTTP comme la majorité des protocoles réseaux attribue aux données expédiées un en-tête qui contient l'adresse IP de la machine cible, diverses informations sur la machine expéditrice, le mode de transfert et le format des données (Figure 9). La connexion est effectuée sur la couche TCP. La version 1.1 de HTTP autorise une connexion persistante alors qu'avec les versions antérieures le serveur fermait la connexion après avoir expédié sa réponse.

3.2.1.2.3.1 Le format d'une requête HTTP

Voici un exemple de requête HTTP d'un client MPSA à un serveur hébergé par la machine « pbil.ibcp.fr » :

```
POST /cgi-bin/mpsa_sec.pl HTTP/1.1\r\n
Host: pbil.ibcp.fr\r\n
User-Agent: MPSA/1.0 (Macintosh; I)\r\n
Content-type: application/x-www-form-urlencoded\r\n
Content-length: 1230d\r\n
\r\n
seq=ATCHKLMSAPCSQR[…]
\r\n
```

La première ligne indique quelle méthode utiliser (*POST*), à quel document l'appliquer (*/cgi-bin/mpsa_sec.pl*) et quelle est la version d'HTTP utilisée par le client (*HTTP/1.1*). La deuxième ligne indique l'adresse IP littérale du serveur à l'aide de la commande *Host*. La commande *User-Agent*

[33] Spécifications protocole HTTP : http://www.w3.org/Protocols

spécifie le nom du client. Ces deux commandes font partie de l'en-tête de requête-client (Figure 9). Les commandes *Content-type* et *Content-length* font partie de l'en-tête d'entité. Elles indiquent, respectivement, le format d'encodage des données et la longueur de ces données.

Une ligne blanche sépare l'en-tête du corps de la requête qui contient les données. Et une autre ligne blanche indique la fin de la requête.

3.2.1.2.3.2 La réponse d'un serveur HTTP

La réponse du serveur différe de la requête principalement par la première ligne et par le format des données lors du transfert.

Tableau 9 : HTTP, les catégories des codes de réponse d'un serveur

Séries de codes	**Signification de la réponse**
100-199	Informationnelle
200-299	Requête client réussie
300-399	Requête client redirigée, autre action nécessaire
400-499	Requête client incomplète
500-599	Erreurs du serveur

Pour la réponse, la première ligne indique la version d'HTTP utilisée par le serveur, le code du statut de la requête et la raison de ce statut sous une forme littérale. Les codes de réponse sont classés en différentes catégories de 100 à 599 (Tableau 9). Les raisons littérales sont par exemple « OK » pour le code 200 ou « Internal Server Error » pour le code 500.

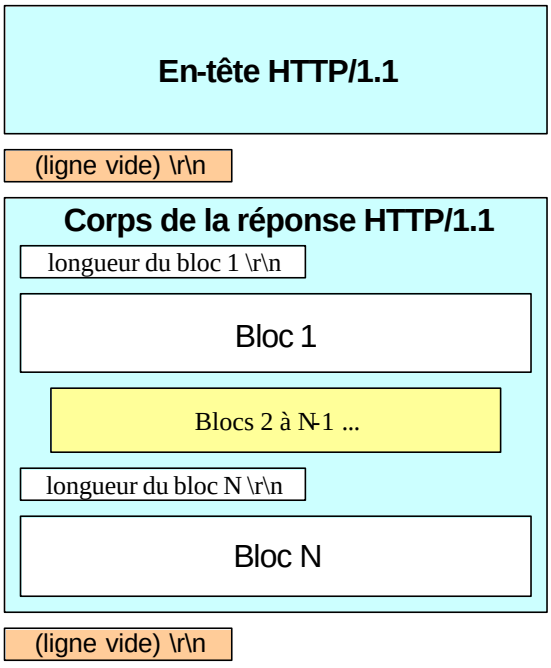


Figure 10 : HTTP, la structure d'une réponse au format « chunked »

Le seul format disponible pour transférer les données lors d'une réponse d'un serveur HTTP/1.1 est le format « chunked ». Ce format est indiqué par la commande HTTP *Transfert-encoding*. Les données sont séparées en une série de tronçons (Figure 10). Ces blocs sont expédiés à la

suite de l'en-tête, avec une ligne blanche intercalée. Chaque bloc est précédé de sa longueur en octet codée en base hexadécimale. Le dernier bloc est suivi d'une ligne vide.

3.2.2 La bibliothèque socket

Lorsque le biologiste se connecte sur notre serveur Web NPS@ à l'aide de son navigateur Web (Nescape, Internet Explorer,…), la connexion TCP/IP est établie par son navigateur. Mais lorsque le biologiste utilise MPSA, nous devons créer cette connexion avant d'échanger les données.

La bibliothèque socket utilise TCP/IP pour établir un lien entre deux applications sur des machines différentes en réseau : le client et le serveur. Cette connexion suit les schémas client-serveur réseau TCP/IP traditionnels, où un client fait une requête réseau sur un port donné et le serveur qui « veillait » sur ce port s'active et répond au client. Cette connexion est faite entre l'application cliente et l'application serveur[34] à travers un tube où transitent les données comme c'est le cas pour l'opérateur « pipe » ('|') entre deux commandes d'un « shell » sous UNIX[35]. La connexion entre le client et le serveur suit un schéma classique (Figure 11) : création de l'environnement, connexion, échange des données, fermeture de la connexion.

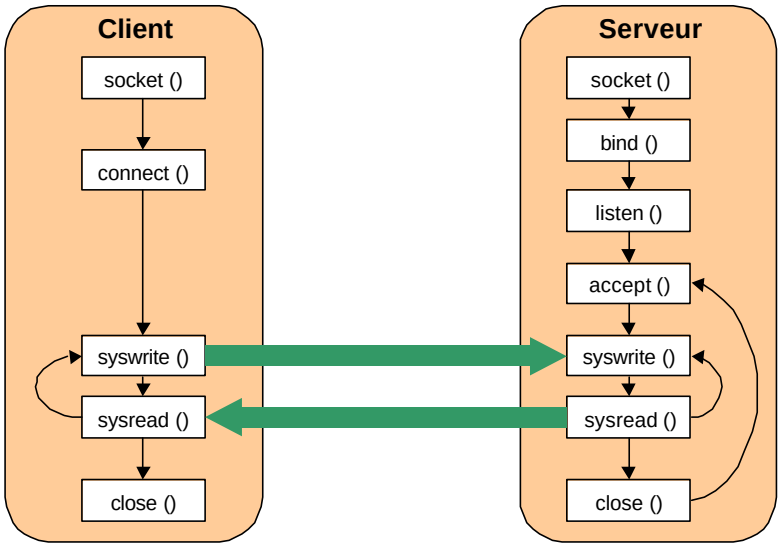


Figure 11 : Schéma d'une connexion entre un client et un serveur à l'aide de la bibliothèque socket.

L'envoi et la récupération des données se font sur cette socket ainsi établie. Pour cela la bibliothèque socket propose des fonctions d'entrées-sorties (E/S) de bas niveau utilisant un tampon (« buffer ») (Tableau 10). La connexion peut être rompue par le client ou le serveur. Ce qui autorise par exemple le mode de connexion « keep-alive » du protocole HTTP. Et généralement c'est le client qui rompt la communication ; le serveur ne rompt la connexion qu'en cas d'erreur comme par exemple une tentative de connexion de clients non-autorisés.

Le client crée le tampon (*socket*), y associe le serveur par une connexion à travers un port donné (*connect*), en l'occurrence le port 80 pour le Web. Client et serveur dialoguent et l'un des deux rompt la connexion (*close*). Bien sûr cela suppose que de son côté le serveur soit opérationnel.

[34] généralement appelé démon (« daemon »).
[35] Par exemple « gzip –dc archive.tar.gz | tar –xvf – » pour décompresser et désarchiver à la volée.

Le serveur devra donc lui aussi créer un tampon générique d'E/S (*socket ()*), associer ce tampon à un port de la machine et « écouter » sur ce port. En effet les applications serveurs sont généralement des tâches de fonds, ou démons. Elles sont le plus souvent initialisées automatiquement lors de la mise en marche de la machine hôte. Ces démons « écoutent » sur un port réseau de la machine. Par exemple dans le cas qui nous concerne, le démon serveur Web est l'application Apache 1.3.3 (P 122, URL 3) qui écoute sur le port 80 de la machine hôte « bmw.ibcp.fr ». Lorsque des données arrivent sur ce port en provenance d'un client, alors le démon s'active[36] lit les données (*sysread ()*), les traite et écrit le résultat sur la socket (*syswrite ()*).

Tableau 10 : Liste des appels proposés par la bibliothèque socket pour connecter deux machines.

Fonction	**Usage**	**Objectif**
socket()	Client et serveur	Créer un tampon générique E/S pour les entrées-sorties dans le système d'exploitation
connect()	Client seul	Etablir une connexion réseau et l'associer au tampon E/S créé par *socket()*
sysread()	Client et serveur	Lire des données de la connexion réseau
syswrite()	Client et serveur	Écrire des données sur la connexion réseau
close()	Client et serveur	Mettre fin à la communication
bind()	Serveur seul	Associer un tampon de socket avec un port de la machine
listen()	Serveur seul	Attendre une connexion entrante d'un client
accept()	Serveur seul	Accepter la connexion entrante du client

3.2.3 Types MIME

MIME[37] est apparu dans le monde de l'Internet comme une extension du protocole SMTP. En effet SMTP avait été prévu pour ne transférer que des fichiers textes. Avec l'apparition du multimédia, le besoin s'est fait ressentir d'échanger aussi des images, des sons, des fichiers compressés. Avec MIME, il est désormais possible de référencer les données échangées en fonction de leur nature et de leur format. Cette méthode simple et efficace a été adoptée par les inventeurs du Web. Ainsi lors d'échanges de données utilisant le protocole HTTP, il est nécessaire de spécifier le format des données transmises pour que l'application qui les reçoit sache comment les traiter.

Tableau 11 : Quelques types MIME et quelques uns de leurs sous-types.

Type	**Sous-type**
text	plain html richtext …
image	jpeg gif tiff …

[36] Et crée généralement des démons fils pour le remplacer puisqu'il devient occupé.
[37] RFC 1341

Type	**Sous-type**
video	mpeg quicktime …
application	postscript rtf mac-binhex40 msword …

La commande sur laquelle s'appuie tout le mécanisme dans HTTP est *Content-type*. Elle indique la nature du fichier transmis. La liste des différents types possibles a été normalisée. Chaque type MIME est défini par l'association d'un type général (image, son, vidéo, texte,…) et d'un sous-type qui indique le format exact des données (Tableau 11). Pour une image par exemple, le sous-type indique s'il s'agit d'un fichier au format gif, jpeg ou d'un autre format. De la même manière les pages Web reçoivent dans leur en-tête le type MIME « text/html » avant d'être expédiées par un serveur à un navigateur Web. Voici comment un serveur Web expédie une page Web :

```
Content-type : text/html

<HTML><HEAD>
<TITLE>Bienvenue à l'IBCP</TITLE>
</HEAD><BODY>
…
</BODY></HTML>
```

Il est important de souligner le retour à la ligne à la fin de la directive *Content-type* qui est nécessaire, ainsi que la ligne blanche qui suit, alors que tous les autres retours à la ligne sont facultatifs. De plus ce retour à la ligne n'est pas un simple '\n' comme dans les fichiers textes UNIX ou '\r' pour les Mac et PC[38], mais explicitement les deux caractères suivants : '\r\n'. Ainsi le vrai format d'une directive *Content-type* pour une page Web est la suivante :

```
« Content-type : text/html\r\n\r\n ».
```

3.3 Développement de logiciels en C

> *« Les fonctions partitionnent les gros traitements en tâches plus petites, et elles permettent de construire des programmes à partir de briques déjà écrites, au lieu de recommencer à zéro. Les fonctions bien conçues cachent les détails de leur fonctionnement aux parties du programme qui n'ont pas besoin de les connaître, ce qui clarifie l'ensemble et facilite les modifications ultérieures.*
>
> *[…] Un programme peut se répartir sur un ou plusieurs fichiers sources. On peut compiler ces fichiers sources séparément et les charger ensemble, accompagnés de fonctions déjà compilées, extraites de bibliothèques. »*
>
> Kernighan BW et Ritchie DM (P 117)

Pour de grosses applications, lorsque les fonctions considérées ont des champs d'applications communs ou dans le cadre de fonctions très génériques, la répartition de ces fonctions en unités de

[38] Cette différence du caractère codant le retour est fondamental lors de l'écriture par des scripts CGI de fichiers-résultat destinés à être lus par des routines de la bibliothèque C standard.

traductions indépendantes est recommandée. Cette répartition produit un certain nombre de fichiers sources et de fichiers d'en-tête qui seront tous compilés indépendamment avant d'être reliés en une application ou simplement regroupés dans une bibliothèque.

Tous ces processus sont réalisés à l'aide d'outils de développement logiciel qui englobent plusieurs types d'applications allant des outils individuels dédiés à une seule tâche comme les compilateurs, débogueurs, « profilers »,…, aux ateliers de génie logiciel en passant par les environnements intégrés de développement. Nous ne considérerons que les outils dédiés au développement de programme en langage C, en sachant que les grandes lignes de ces schémas sont aussi applicables aux autres langages de 3^{e} génération (L3G) comme le fortran ou le pascal.

3.3.1 Architecture répartie des programmes C.

L'inclusion de fichiers permet la répartition des codes sources d'une application en différentes unités de traduction, dans lesquelles les instructions de l'implémentation seront regroupées par champ d'application (Figure 12).

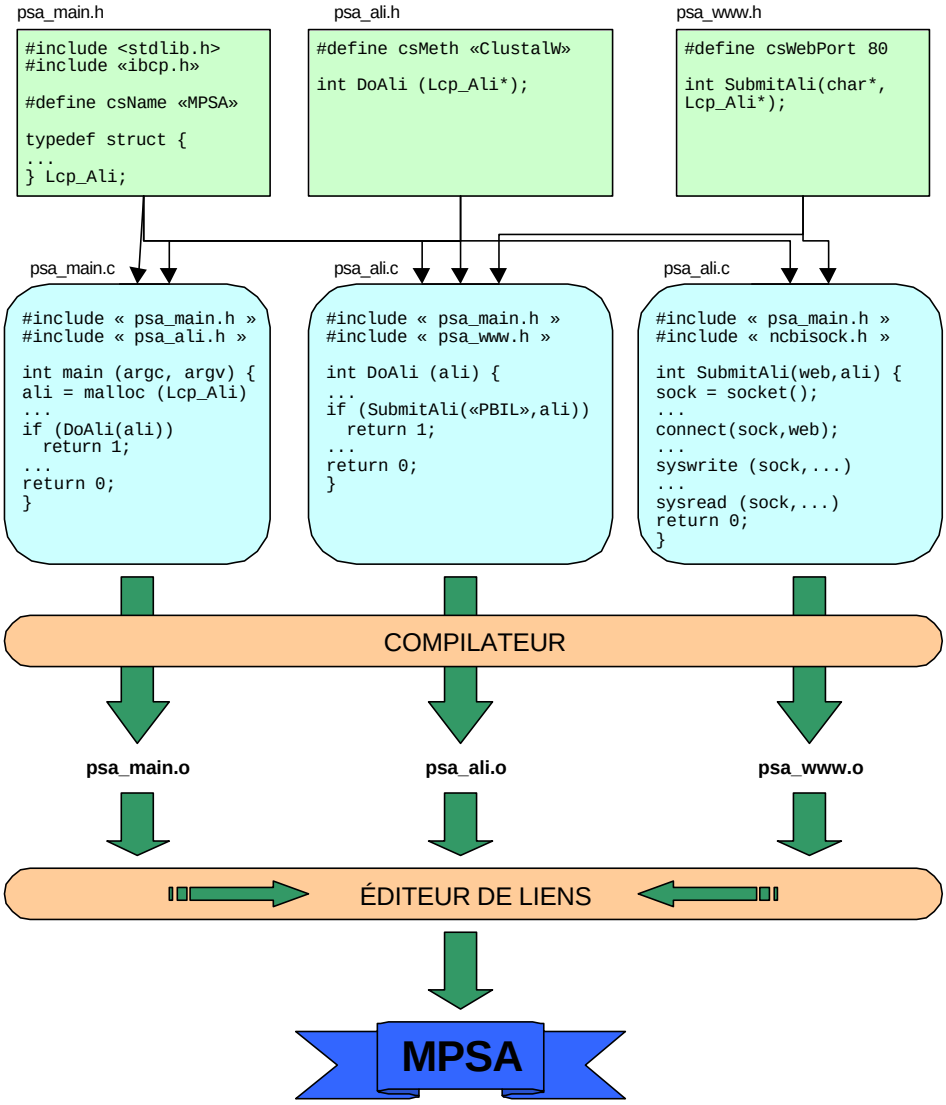


Figure 12 : Organisation des unités de traduction d'un programme C. Exemple du logiciel MPSA.

Les fichiers sources C regroupent les jeux d'instructions définissant les fonctions. C'est également dans ces fichiers sources que sont définies et initialisées les différentes variables utiles à l'implémentation. Les fonctions, tout comme les variables, sont classées en deux catégories, *static* ou *extern*, suivant leur portée.

Les « headers » ou fichiers d'en-tête permettent de rassembler dans un fichier des définitions générales utiles à plusieurs fichiers. Ce peuvent être des lexèmes[39] , des déclarations de constantes, de variables ou de fonctions définies et initialisées dans d'autres fichiers sources.

3.3.2 Le préprocesseur C

Le préprocesseur du C est une particularité par rapport aux autres langages compilés qui n'en possèdent pas tous. Ce préprocesseur est une première étape, distincte, de la compilation. Il offre certaines possibilités syntaxiques comme les inclusions de fichiers, la définition de lexèmes ou la compilation conditionnelle.

3.3.2.1 L'inclusion de fichiers et les lexèmes

Les deux instructions employées pour l'inclusion de fichiers et la définition de lexèmes sont, respectivement, *#include* et *#define*. La première permet d'inclure le contenu d'un autre fichier au moment de la compilation, la seconde de définir un lexème qui sera remplacé avant la compilation par une chaine arbitraire de caractères.

La ligne suivante permet d'insérer la déclaration des variables et fonctions propres à la bibliothèque socket (*cf.* 3.2.1.2.3) du NCBI :

```
#include <ncbisock.h>
```

Les lexèmes permettent de définir des constantes symboliques lorsque la chaine de remplacement est une valeur d'un type quelconque, et de définir également des macros lorsque la chaine de remplacement est un jeu d'instructions. La déclaration suivante définit la constante symbolique contenant le numéro du port réseau utilisé par le Web :

```
#define csWebPort 80
```

et celle-ci la macro retournant le plus grand de deux nombres a et b :

```
#define MAXIMUM(a,b) ((a) > (b) ? (a) : (b))
```

3.3.2.2 La compilation conditionnelle

Le préprocesseur C possède également des structures de contrôles permettant de faire de la compilation conditionnelle. Ces structures sont du type :

Si (condition) {…}

[sinon si (condition) {…}]

[sinon {…}]

fin si

Ces structures de contrôle suppriment purement et simplement le code sous leur dépendance si la condition n'est pas remplie. Cette compilation conditionnelle est généralement utilisée pour réaliser

[39] Ou *token*

des implémentations différentes suivant l'architecture logicielle sur laquelle est compilée l'application. La condition à réaliser porte soit sur l'existence d'une constante conditionnelle, « *#ifdef csConstanteSymbolique* », ou sur la valeur d'une constante « *#if (csConstanteSymbolique > 20)* ».

3.3.3 Compilateur et éditeur de liens

Le compilateur intègre en fait le préprocesseur. Un exemple de compilateur C est 'cc' sous UNIX, mais il en existe d'autres plus efficaces comme 'gcc' qui fait partie de l'ensemble logiciel GNU (URL 21). Le compilateur analyse les programmes sources et convertit le code de haut niveau syntaxique qu'est le langage C en instructions de bas niveau propre à l'architecture du processeur de la machine. Cette compilation se fait suivant un degré d'optimisation choisi par le programmeur. Chaque unité de traduction produit un fichier binaire en langage de bas niveau.

Ces binaires (**.o*) peuvent ensuite être archivés dans une bibliothèque. Ils ne subissent alors pas de liaison. Par contre, pour la compilation d'un logiciel les binaires doivent être liés. L'étape d'édition des liens fait coincider les objets ou fonctions de mêmes noms à l'intérieur d'une même unité de traduction si ces objets ou fonctions sont identifiés comme internes, ou entre les différentes unités de traduction si celles-ci sont identifiés comme externes. Cette faculté de créer les liens entre des unités de traduction différentes permet d'utiliser des bibliothèques compilées précédemment par le programmeur ou distribuées par d'autres programmeurs.

3.3.4 L'outil « make »

L'outil make d'UNIX est un gestionnaire de compilation proposant une mise en place simple des relations de dépendance entre les unités de traduction et une gestion efficace des options de compilation. La commande make utilise un script de définition des règles de compilation. Par défaut le script utilisé est le fichier « makefile » du répertoire courant (*cf.* l'Annexe B) contenant le makefile fixant les règles de compilation de MPSA).

Le programmeur peut définir des variables à l'intérieur d'un script générique et les modifier directement depuis la ligne de commande pour effectuer des compilations spécifiques. Le script contient également une ou plusieurs cibles, qui peuvent dépendre les unes des autres. Et l'utilisateur choisit lors de l'exécution du script quelle cible développer. Ces fonctionnalités permettent de construire en une seule session différentes applications ayant une ou plusieurs unités de traduction en commun.

3.3.5 Débogueur symbolique : dbx

Un débogueur symbolique permet de corriger les bogues se produisant lors de l'exécution d'un programme. Un exemple est le débogueur symbolique d'UNIX : dbx. Le programme doit avoir été compilé avec les options de compilations adéquates pour pouvoir être débogué, il s'agit par exemple de l'option '-g' avec le compilateur « cc » sous UNIX. Une table des symboles est alors générée et décrit le contexte pour chaque instruction élémentaire et espace mémoire utilisés.

Le débogueur symbolique fait office d'interface entre le programme est l'environnement d'exécution. Il intercepte les signaux d'erreurs et indique au programmeur le type de l'erreur et sa position dans le code source. Le programmeur peut de plus afficher la valeur des variables utilisées à un instant t. Le débogueur symbolique propose également une exécution pas à pas ou conditionnelle du logiciel.

3.3.6 Environnement de développement : CodeWarrior® et Visual C++®

Les environnements de développement intègrent tous les outils nécessaires de l'écriture du code source à son évaluation dans un profiler ou à son déboguage. Ils proposent un éditeur de texte proposant un codage en couleur de la syntaxe. De nombreuses fonctionnalités comme la liste des fichiers d'en-tête utilisés ou celle des fonctions définies dans les sources permettent d'avoir un aperçu global du programme. Ils incluent également le compilateur et le débogueur et proposent des liaisons entre les fenêtres et les données des différents outils. Le programmeur peut ainsi passer d'une donnée à une autre ayant une relation avec la précédente sans passer par de nombreuses manipulations. Par exemple un double clic sur le compte-rendu d'une erreur de compilation ouvre le fichier source correspondant et le propose en mode édition.

Nous utilisons les environnements de développement Metrowerks CodeWarrior Professional® Release 4 et Microsoft Visual C++® Professional Edition Version 5.0.

3.4 Langages

Pour nos logiciels et nos services nous employons différents langages pour la description des opérations ou pour la mise en page de documents.

3.4.1 Langages de développement

Les deux langages de développement que nous avons employés sont le langage C, qui est compilé avant exécution, et le langage Perl, qui est interprété lors de l'exécution.

3.4.1.1 Le langage C norme ANSI

Le langage C est un langage de 3^{e} génération développé par Denis Ritchie au début des années soixante-dix dans les laboratoires AT&T Bell (P 117, URL 4). Il fut normalisé de 1983 à 1988 par le comité X3J11 de l'Institut National Américain de Standardisation (ANSI) pour aboutir au langage C_{ANSI} tel qu'il est défini de nos jours. Le langage C n'est pas lié à une architecture matérielle ou à une machine particulière. Un programme écrit en C fonctionne sur n'importe quelle machine l'acceptant, sans qu'il soit nécesssaire de modifier le code.

Le langage C est un langage compilé, typé et structuré. Une gestion dynamique de la mémoire est accessible au travers d'un jeu d'instructions qui gère une table d'allocation de la mémoire. Ainsi une application implémentée de façon adéquate fait une gestion efficace de la mémoire qu'elle utilise. Il n'est donc pas nécessaire de réserver lors de l'écriture des fichiers sources de grands éléments de mémoire pour pouvoir traiter de grandes quantités de données.

Un des éléments qui apportent une grande puissance au langage C est le pointeur avec la souplesse et les possibilités qu'il propose. Le pointeur peut référencer n'importe quel type d'espace mémoire et même ne pas avoir de type explicite puisqu'on peut pointer sur le type *void.* Il est alors possible d'écrire des routines efficaces qui transfèrent les variables sans se soucier de leur type jusqu'à la fonction nécessaire au traitement. Il est également possible de définir des pointeurs de fonctions génériques qui prennent leur valeur en fonction du contexte, et permettent l'exécution d'une fonction précise sans avoir à utiliser des structures de contrôle à chaque appel.

3.4.1.2 Perl

Perl est un langage conçu initialement pour traiter de grandes quantités de texte et rédiger des rapports sous UNIX (P 184, URL 39). Ses différentes évolutions en ont fait un langage qui permet de manipuler facilement non seulement du texte, mais aussi des systèmes de fichiers et des processus. C'est un langage interprété disponible gratuitement pour toutes les architectures matérielles et logicielles importantes.

Grâce à ses capacités à manipuler des fichiers et des processus, à sa souplesse et à sa sécurité d'utilisation, Perl est devenu le langage de prédilection pour les concepteurs d'applications CGI (en interaction avec le Web) et pour les administrateurs systèmes et réseaux : c'est-à-dire toutes les personnes qui conçoivent des applications pour automatiser des processus et mettre à jour de grandes quantités de fichiers, qu'ils soient locaux ou accessibles par le réseau. Par exemple le chargement de toutes les données d'un fichier est réalisé en deux instructions avec Perl alors qu'il en faudrait plusieurs dizaines en langage C :

```
open fp_in, «<nomfichier» ;
@tampon = <fp_in> ;
```

Les expresssions rationnelles (*regex*[40]) sont utilisées pour donner une description, un modèle (« pattern »), une forme générale d'une séquence de caractères que l'on recherche dans une chaine, sans avoir à préciser la séquence entièrement. Il y a concordance si la *regex* répond affirmativement à la question suivante : « Trouve-t-on une séquence X dans la chaine Y ? ». Le mécanisme des expressions rationnelles n'est pas spécifique à Perl. Mais il y prend tout son sens car il se trouve alors associé à la grande puissance de manipulation de texte de Perl et au mécanisme de substitution automatique.

Un exemple très simple d'expression rationnelle est la création de liens hypertextes dans un fichier contenant des URL. Il suffit de charger en mémoire le fichier avec les deux expressions vues ci-dessus. L'encadrement de tous les URL avec les balises HTML indiquant un lien hypertexte est réalisé en une seule instruction :

```
@tampon =~ s/(\w+:\/\/[a-z0-9\/:.-_]+)/<A href=$1>$1<\/A>/gimos;
```

De plus Perl est distribué à travers le CPAN[41] qui propose de nombreux modules regroupant les fonctions par champs d'application. Notamment les modules CGI et LWP constituent une interface au traitement des requêtes CGI et permettent des connexions réseau efficaces. Par exemple le module CGI propose un objet global qui effectue automatiquement la séparation des différents arguments d'un script CGI et les stocke dans un hachage :

```
$html = new CGI ;
print $html->param ('seq'), "\n" ;
```

La première ligne crée le nouvel objet de type CGI et le stocke dans la variable *$html*. La seconde ligne affiche sur la sortie standard la valeur du paramètre 'seq'.

[40] regex : REGular Expression.
[41] CPAN : Comprehensive Perl Archive Network.

3.4.2 Langages de mise en forme

Après les langages de développement, nous décrivons les langages graphiques que nous utilisons pour nos applications : PostScript, RTF et HTML.

Les langages de mise en forme permettent de coder par une syntaxe spécifique des éléments graphiques (droite, point, rectangle, couleur de trait,…) et la mise en page du texte (attribut de paragraphe, casse de la police, couleur de trait,…).

Les scripts peuvent être écrits avec ces langages par un programmeur directement à l'aide d'un simple éditeur de texte, mais c'est lourd et sans grand intérêt. Les scripts de mise en forme sont majoritairement écrits par des applications qui proposent ainsi une sortie de qualité à leurs utilisateurs. Il s'agit alors de programmation indirecte puisque le programmeur implémente avec son langage de développement le logiciel et la mise en forme des données avec le langage adéquat.

3.4.2.1 PostScript

PostScript (PS) est un langage développé par Adobe Systems Inc. pour la description de documents graphiques destinés à l'impression (P 154). En fait c'est un langage de description de pages. La description des éléments graphiques de chaque page du document est réalisée vectoriellement à l'aide de primitives de dessin ou de mise en forme. Le fichier obtenu est un script PS qui est ensuite téléchargé sur une imprimante compatible. Une imprimante est compatible PS si elle possède un interpréteur PS câblé ou en émulation.

Postscript un langage procédural typé. Les procédures peuvent être regardées soit comme des objets, soit comme des opérateurs composites. La gestion de la mémoire est organisée en plusieurs dictionnaires fonctionnant suivant la méthode de la pile LIFO (Last-In First-Out). La rédaction des instructions utilise la notation polonaise inverse[42]. Ainsi les opérateurs piochent leur argument sur la pile qui devra avoir été remplie précédemment par la déclaration de nouvelles valeurs ou par les résultats des précédents opérateurs.

Un exemple d'une routine écrite sous la forme d'une macro est la suivante qui décrit le dessin d'un rectangle aux coins arrrondis :

```
% Dessin d'un rectangle aux coins arrondis (rayon r), largeur 2a hauteur 2b
% centre en (x,y)
% Parameters: x y a b r
/rectar  {
/r exch def /b exch def /a exch def gsave
translate a neg 0 2 copy moveto
a neg b neg a b neg a b a neg b
4 { 3 index 3 index r arcto 4 { pop } repeat } repeat
a neg 0 lineto closepath
stroke pop pop grestore
} def
```

3.4.2.2 RTF

RTF est un langage de mise en forme de texte développé par Microsoft (P 160). Cette mise en forme est réalisée au moyen de balises (Tableau 12). Ces balises modifient les attributs des textes ou

[42] Les opérandes précèdent toujours les opérateurs.

objets sous leur dépendance. Les attributs modifiés sont par exemple la police employée, sa taille, sa casse, la couleur du trait, le style du paragraphe, *etc*.

Un fichier RTF est un simple fichier texte (ASCII) qui contient des données et leur mise en forme graphique. Les documents obtenus peuvent être incorporés dans n'importe quel éditeur de texte (Microsoft Word, Claris Works,…) puisque ce format est devenu un standard de format de texte mise en forme.

Tableau 12 : Quelques éléments du langage RTF

Balise RTF	**Utilité**
\redN	Valeur de la composante rouge en mode RGB[43]
\greenN	Valeur de la composante verte en mode RGB
\blueN	Valeur de la composante bleue en mode RGB
\title	Référence le titre du document
\par	Indique la fin d'un paragraphe
\pard	Initialise un paragraphe avec les propriétés par défaut
\fN	Identifie le numéro de la police
\fsN	Positionne la taille de la police en demi-points
\landscape	Orientation à l'italienne
\margl	La marge gauche
\margr	La marge droite

3.4.2.3 HTML

HTML[44] est un langage de mise en forme de documents hypertextes destinés au Web. Il utilise lui aussi des balises (Tableau 13) pour mettre en forme le texte, insérer des images, créer des tableaux mais également créer des liens hypertextes sur d'autres documents. Ces liens hypertextes sont des zones actives du document qui réagissent lors d'un clic de souris. Alors le document référencé par ce lien est rapatrié depuis la machine qui l'héberge et affiché sur la machine locale à la place du précédent (P 35).

Tableau 13 : Quelques balises du langage HTML

Balises HTML	**Utilité**
<HTML>…</HTML>	Délimite un document HTML
<H1>…</H1>	Délimite un titre de 1er niveau : le plus haut
<H7>…</H7>	Délimite un titre de 7^{e} niveau : le plus bas
<UL>…</UL>	Délimite une liste non ordonnée
<LI>…</LI>	Délimite une entrée d'une liste
<TABLE>…</TABLE >	Délimite un tableau
<G>…</G>	Affiche le texte encadré en gras

[43] RGB : Red Green Blue. Mode de couleur additif. Chaque couleur a 3 composantes (rouge, verte, bleue) prenant une valeur de 0 à 255.
[44] Actuellement dans sa version 4.0. Il est défini par la RFC 1866.

Balises HTML	**Utilité**
<I>…</I>	Affiche le texte encadré en italique

3.5 Bibliothèques de fonctions

Nous avons présenté les différents langages que nous utilisons pour nos développements. Mais nous employons également les développements proposés par d'autre programmeurs. Ces implémentations sont distribués sous la forme de bibliothèques regroupant plusieurs routines le plus souvent relatives aux mêmes champs d'application.

Nous évoquons ici les trois bibliothèques en langage C utilisées pour le développement de MPSA : la bibliothèque standard C, Vibrant et ReadSeq.

3.5.1 La bibliothèque standard C

La bibliothèque standard C (P 155) est l'outil indispensable de tous développeurs de programme en C. Elle regroupe de nombreuses fonctions essentielles aux développements courants comme la gestion des entrées/sorties, la gestion de la mémoire, le traitement des chaines,… Le comportement de ces fonctions est normalisé quelle que soit l'architecture matérielle de la machine.

Les différentes fonctions sont regroupées par champ d'application. Et leurs déclarations sont regroupées en fonction des thèmes dans différents fichiers d'en-tête (Tableau 14).

Tableau 14 : Les fichiers d'en-tête standards de la bibliothèque C (d'ap. P 155).

Fichier d'en-tête	**Champ d'application**
assert.h	Mise en œuvre d'assertions à des endroits cruciaux d'un programme
ctype.h	Test et conversion des caractères
errno.h	Détection des conditions d'erreur
float.h	Limites et paramètres relatifs aux nombres à virgule flottante
limits.h	Définition des limites et de différents paramètres propres au système
locale.h	Adaptations locales pour traiter les problèmes d'internationalisation (ou *I18N*[45])
math.h	Déclaration des fonctions mathématiques
setjmp.h	Court-circuit du déroulement normal des appels et retours de la fonction
signal.h	Traitements des signaux (synchrones et asynchrones)
stdarg.h	Gestion des listes variables d'arguments des fonctions
stddef.h	Définitions standards
stdio.h	Gestion et manipulation des entrées/sorties
stdlib.h	Fonctions d'utilité générale[46]
string.h	Manipulation des tableaux de type caractère ainsi que d'autres objets

[45] 18 est le nombre de lettres qui sépare le I du N dans le mot « internationalisation ».
[46] stdlib.h est un peu le fourre-tout de la bibliothèque C.

Fichier d'en-tête	Champ d'application
	considérés comme tels
time.h	Manipulation du temps

3.5.2 La bibliothèque Vibrant

La bibliothèque Vibrant est une bibliothèque d'interface utilisateur graphique (GUI) à vocation biologique, de haut-niveau et multiplateforme. Elle est développée par l'équipe du Dr Jonathan Kans au NCBI (URL 30) et est utilisée par les chercheurs du NCBI pour construire l'interface graphique des outils bioinformatiques qu'ils développent : Entrez[47], Sequin[48], C3nd[49].

Cette bibliothèque GUI est écrite en langage C et propose des fonctions graphiques indépendantes d'une architecture logicielle particulière. Elle permet ainsi de concevoir des interfaces graphiques « portables » d'un système d'exploitation à un autre. Les fonctionnalités graphiques proposées, appelées *widget*, sont de différents types : différentes variétés de fenêtres, des menus déroulants, des boutons poussoirs/à choix multiples/à choix exclusif, des plans graphiques de dessin, des ascenseurs, *etc.* La bibliothèque Vibrant s'occupe de la gestion des « évènements », c'est-à-dire « un choix a été fait dans un menu déroulant », « un clic a été effectué avec la souris à un endroit donné »,…. A charge du programmeur de mettre en place toutes les actions à effectuer en fonction de ces événements.

La bibliothèque Vibrant assure l'interface entre le programmeur et les différentes bibliothèques graphiques des architectures logicielles supportées : MOTIF™ et Xlib sous UNIX (Lesstif sous Linux), ActiveX® sous Microsoft Windows®, les ressources graphiques sous MacOS®. Ce qui veut dire que la bibliothèque Vibrant propose des fonctions pour créer des fenêtres, des boutons ou d'autres widgets[50] :

```
WindoW DocumentWindow (Int2 left, Int2 top, Int2 width,
                       Int2 height, CharPtr title,
                       WndActnProc close, WndActnProc resize);
WindoW FixedWindow    (Int2 left, Int2 top, Int2 width,
                       Int2 height, CharPtr title,
                       WndActnProc close);
WindoW ModalWindow (Int2 left, Int2 top, Int2 width,
                    Int2 height, WndActnProc close);
ButtoN PushButton    (GrouP prnt, CharPtr title, BtnActnProc actn);
ButtoN DefaultButton (GrouP prnt, CharPtr title, BtnActnProc actn);
ButtoN CheckBox      (GrouP prnt, CharPtr title, BtnActnProc actn);
```

La bibliothèque Vibrant se charge ensuite d'invoquer les routines des bibliothèques graphiques correspondant à l'architecture sur laquelle le programme est compilé. Elle limite également le nombre des variables propres à chaque contexte graphique, comme c'est le cas par exemple pour la bibliothèque Xlib où il faut définir et initialiser différentes variables (*GC*, *Display*, *Window root*,…) avant de créer une fenêtre fille.

La bibliothèque Vibrant n'est cependant pas d'un aussi haut niveau que la bibliothèque d'outil Tk développée pour le langage de scripts Tcl (URL 52). Prenons l'exemple du redimensionnement d'une fenêtre. Tk adapte la taille des widgets contenue dans la fenêtre concernée pour qu'ils soient

[47] Entrez : http://www.ncbi.nlm.nih.gov/Entrez/
[48] SequIn : http://www.ncbi.nlm.nih.gov/Sequin/index.html
[49] Cn3D : http://www.ncbi.nlm.nih.gov/Structure/CN3D/cn3d.html
[50] widgets : objets graphiques d'un système de fenêtrage

toujours entièrement visibles. Vibrant ne le fait pas et une partie des widgets peut être ainsi masquée et hors d'atteinte. Ainsi si on considère la fonction *DocumentWindow* qui permet de créer une fenêtre redimensionnable :

```
WindoW DocumentWindow (Int2 left, Int2 top, Int2 width,
                       Int2 height, CharPtr title,
                       WndActnProc close, WndActnProc resize);
```

Il existe bel et bien un champ *resize* mais il ne correspond qu'à la déclaration par l'utilisateur de la fonction de retour, ou « callback », qui sera invoquée par la bibliothèque lors du redimensionnement de la fenêtre. En aucun cas cette gestion des widgets fils d'une fenêtre n'est implémentée dans la bibliothèque Vibrant.

De la même manière l'affichage de texte coloré est relativement lent avec la librarie Vibrant. Ceci est du aux nombreux facteurs du contexte graphique à gérer pour un affichage standard quel que soit l'objet. C'est un exemple où la simplification de l'utilisation va à l'encontre de l'efficacité. En tout cas cette relative lenteur n'est que difficilement conciliable avec par exemple l'affichage d'un bloc d'alignement de 50 séquences de 400 acides aminés en 4 couleurs.

Nous étudierons plus longuement par la suite (*cf.* 4.5.2) les solutions que nous avons implémentées pour passer outre ces deux inconvénients.

3.5.3 ReadSeq

ReadSeq est un programme C de lecture/écriture/conversion de fichiers de séquences aux formats standards (Tableau 15). Il a été écrit par D.G. Gilbert[51] de l'Université d'Indiana. Nous n'utilisons en fait que le fichier source *ureadseq.c*. Nous l'utilisons comme une bibliothèque des fonctions disponibles pour la lecture et l'écriture des séquences aux formats standards. Ces fonctions sont déclarées dans le fichier en-tête *readseq.h* :

```
extern short seqFileFormat(const char *filename, long *skiplines, short *error );
extern short seqFileFormatFp(FILE *fseq, long  *skiplines, short *error );

extern char *listSeqs(const char *filename, const long skiplines,
                      const short format, long *nseq, short *error );

extern char *readSeq(const short whichEntry, const char *filename,
                     const long skiplines, const short format,
                     long *seqlen, long *nseq, short *error, char *seqid );

extern char *readSeqFp(const short whichEntry_, FILE  *fp_,
  const long  skiplines_, const short format_,
        long  *seqlen_, long *nseq_, short *error_, char *seqid_ );

extern short writeSeq(FILE *outf, const char *seq, const long seqlen,
                      const short outform, const char *seqid );
```

La fonction *seqFileFormat* permet de reconnaître le format du fichier considéré (*filename*). Les fonctions *readSeq* et *writeSeq* permettent respectivement de lire et d'écrire des séquences nucléotidiques ou des séquences d'acides aminés dans la plupart des formats utilisés pour les banques de séquences.

Cependant cette bibliothèque comporte certains bogues qui limitent son utilisation : par exemple une séquence écrite avec *writeSeq* dans le format Pearson/FASTA ne sera pas correctement

[51] gilbertd@bio.indiana.edu

relue par un appel ultérieur à *readSeq* (notamment l'identificateur diffère). C'est pourquoi nous recommandons dans MPSA l'utilisation du format EMBL pour l'écriture et la lecture des séquences d'acides aminés.

Tableau 15 : Formats de séquences reconnus par la bibliothèque ReadSeq.

Code du format	Format	Type de séquence
1	IG/Stanford	Acides nucléiques
2	GenBank/GB	Acides nucléiques
3	NBRF	Acides nucléiques
4	EMBL	Acides nucléiques
5	GCG	Mixte[52]
6	DNAStrider	Acides nucléiques
7	Fitch	Acides nucléiques
8	Pearson/Fasta	Mixte
9	Zuker (lecture seulement)	Acides nucléiques
10	Olsen (lecture seulement)	Acides nucléiques
11	Phylip3.2	Acides nucléiques
12	Phylip	Acides nucléiques
13	Plain/Raw	Mixte
14	PIR/CODATA	Protéines
15	MSF	Mixte
16	ASN.1[53]	Divers
17	PAUP	Acides nucléiques
18	Pretty (écriture seulement)	Acides nucléiques

De plus le format Pearson/FASTA condensé n'est pas supporté par ReadSeq. Or les banques de séquences utilisées pour les recherches de similarités sont le plus souvent dans ce format car il permet un stockage de taille réduite plus approprié à des échanges électroniques. Une entrée dans ce format se compose d'une ligne d'identification suivie d'une ou plusieurs lignes contenant la séquence de cette protéine :

```
>nomBanque|ac_général|seq_ident commentaires …\n
sequencedacideamine…\n
suitedelasequencedacideamine…\n
suitedelasequencedacideamine…\n
```

La ligne d'identification est unique mais elle a une longueur variable qui dépend des commentaires. Par contre, les lignes contenant la séquence en acides aminés de la protéine sont en nombre variable mais elles ont une longueur maximale de 80 caractères. De plus la ligne

[52] Acides nucléiques et proteines

[53] Cette syntaxe est utilisée pour la présentation des données sous la forme d'objets entre la couche application et la couche présentation du modèle OSI de l'architecture d'un réseau (Figure 6).

d'identification commence par le caractère '>', celui-ci est suivi de 3 champs séparés par le caractère '|' (« pipe »). Ces champs sont dans l'ordre :

- « *nomBanque* », le nom de la banque en un seul mot,
- « *ac_général* », un code d'accès générique pour la protéine, pas toujours présent, le champ est alors vide (`>nomBanque||seq_ident commentaires …\n`) ,
- « *seq_ident commentaires* », le code d'accès pour la protéine dans la banque considérée suivi de commentaires d'ordre biologique.

4 Résultats et discussion

4.1 Recherche de motifs – « pattern matching »

Dans certains cas la séquence de la protéine analysée est trop éloignée de toutes celles déjà identifiées pour qu'il soit possible de détecter une ressemblance à l'aide des outils de recherche de similarité. Cependant une relation peut être mise en évidence par l'existence d'une même disposition de certains acides aminés dans la séquence. Mais l'augmentation de la taille des familles de protéines provoque une diminution de la spécificité des motifs protéiques connus. En effet il arrive fréquemment que certaines positions des motifs deviennent trop dégénérées, ne soient alors plus suffisamment discriminantes lors d'une recherche et provoquent l'apparition de faux-positifs.

C'est pourquoi la définition d'un motif protéique doit permettre de trouver toutes les occurrences vérifiées expérimentalement et minorer le nombre de faux-positifs. De plus le motif défini sera utilisé sur les nouvelles séquences protéiques publiées pour y rechercher de nouvelles occurrences. Et lors de cette recherche de motifs potentiels, il est fréquent d'autoriser un certain taux d'erreur afin de ne pas oublier des motifs qui ne correspondent pas mais qui cependant peuvent faire partie de la même famille de protéines.

C'est pourquoi nous avons mis au point une recherche de motifs protéiques qui privilégie les positions *a priori* biologiquement les plus importantes du motif, c'est-à-dire les positions les plus strictes.

4.1.1 Syntaxe PROSITE

Nous utilisons la syntaxe de Prosite (P 104) pour la définition des motifs protéiques recherchés. Cette syntaxe s'applique non seulement pour l'écriture du motif mais également pour le format du fichier contenant ce motif.

Tableau 16 : Les identificateurs de ligne d'une entrée Prosite

Identificateur	Signification de la ligne
ID	Première ligne d'une entrée
AC	Numéro de l'entrée
DT	Date
DE	Brève description
PA	Motif
MA	Matrice/profil
RU	Règles (« rules »)
NR	Résultats numériques
CC	Commentaires
DR	Références croisées vers SWISS-PROT
3D	Références croisées vers la PDB

Identificateur	Signification de la ligne
DO	Numéro de l'entrée dans la documentation

Le format d'une entrée d'un fichier Prosite est organisé en plusieurs lignes repérées par un identificateur à deux lettres (Tableau 16). Il est possible d'utiliser pour notre méthode une entrée Prosite complète mais seulement trois champs sont nécessaires : ID, DE et PA. Ainsi le format d'un fichier de motif aura la structure minimale suivante :

```
//
ID   (une seule ligne)
DE   (une seule ligne)
PA   (autant de lignes de 78 caractères nécessaires pour definir le motif)
//
```

La définition du motif dans les lignes PA devra respecter les conventions suivantes :

- Utilisation du code à une lettre IUPAC (URL 24) pour les acides aminés,
- Le symbole 'x' est utilisé pour une position où tous les acides aminés sont autorisés,
- Les ambiguités sont indiquées en répertoriant entre crochets (« [] ») tous les acides aminés autorisés à une position donnée (*e.g.* [AM] indique que Ala et Met sont autorisés),
- Les ambiguités sont également indiquées en répertoriant entre accolades (« {} ») tous les acides aminés exclus à une position donnée (*e.g.* {AM} indique que Ala et Met sont exclus),
- Chaque position est séparée de la suivante par un tiret '-',
- Une position répétée est suivie par le nombre de ses répétitions indiqué entre parenthèses, ce peut être un nombre fixe ou un intervalle de variation (*e.g.* A(3) signifie A-A-A ; [DE](1,2) signifie [DE] ou [DE]-[DE]),
- Lorsque les occurrences d'un motif sont limitées aux extrémités N- ou C-terminales, alors la définition du motif est précédée de '<' ou suivie de '>',
- Un motif est terminé par un point.

4.1.2 Principe

L'originalité de notre méthode repose sur une répartition variable de l'information biologique entre les différentes positions du motif recherché. Pour cela nous pénalisons l'absence d'une position en fonction de sa fréquence. Ainsi une position avec beaucoup d'ambiguités sera moins discrimante lors d'une recherche qu'une position moins ambiguë, et *a fortiori* qu'une position stricte.

La première étape consiste à calculer le score de chaque position du motif et le score global du motif. La fréquence du motif F_t est utilisée comme valeur initiale du score global du motif Sc_0.

Nous utilisons pour calculer le score de chaque position du motif, les fréquences des acides aminés relevées dans SWISS-PROT. Chaque position i se voit alors attribuer sa fréquence F_i comme score de pénalisation P_i. Nous considérons une ambiguité de présence de la même manière qu'une ambiguité d'exclusion. Ainsi la position '[FYW]' aura le même score de pénalisation que '{FYW}'. Ce score de pénalisation sera utilisé lorsque cette position est absente du motif potentiel.

Le biologiste définit comme critère de recherche le taux minimal τ d'information biologique que les motifs retenus doivent contenir par rapport au motif recherché. À l'aide de ce taux minimal et du score initial Sc_0 nous définissons le score seuil Sc_{seuil} du motif recherché :

$$Sc_{seuil} = Sc_0^{\tau}$$

Ce score seuil Sc_{seuil} définit alors avec le score initial Sc_0 l'intervalle de variation autorisé pour le score global du motif (Figure 13).

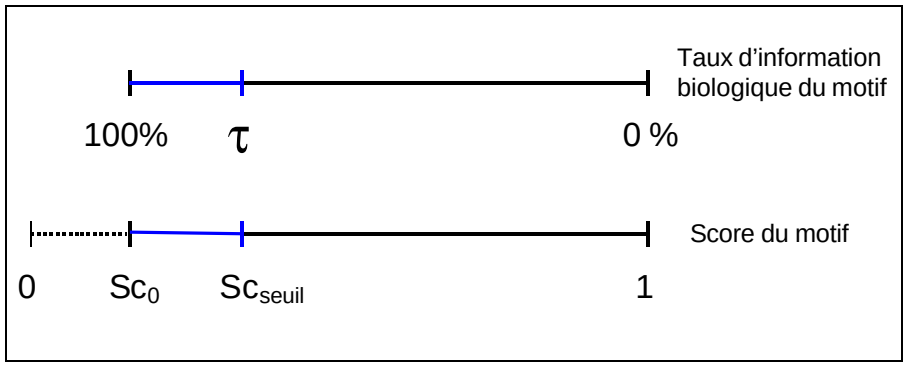


Figure 13 : Détermination de l'intervalle de variation du score du motif recherché

Ensuite nous recherchons le motif dans la séquence de la protéine de longueur m. Pour cela nous déplaçons, de l'extrémité N-terminale à l'extrémité C-terminale, une fenêtre égale à la longueur du motif. Pour chaque position de cette fenêtre, nous comparons l'acide aminé i de la séquence à la position i du motif, avec i variant de 1 à la longueur n du motif. Si cet acide aminé i correspond au motif, alors le score Sc_i du motif est le même que le score de la position précédente Sc_{i-1}. À l'inverse si l'acide aminé ne correspond pas, alors le score Sc_i est égal au score précédent Sc_{i-1} divisé par le score de pénalisation P_i de la position i du motif recherché.

Si toutes les positions de la fenêtre et du motif ont été comparées sans que la valeur du score n'ait quitté l'intervalle autorisé $[Sc_o ; Sc_{seuil}]$, alors le motif est retenu.

4.1.3 Algorithme

```
/*
Initialisation des scores de pénalisation des n positions du motif
*/
Pour i=1 à n Faire
      Si présence de {} Faire
      Pi = somme des fréquences des acides aminés EXCLUS
      sinon
      Pi = somme des fréquences des acides aminés AUTORISÉS

/*
Initialisation du score initial Sc0 du motif
*/
Sc0 = 0
Pour i=1 à n Faire
      Sc0 *= Pi

/*
Initialisation du score seuil Scseuil du motif pour τ=80%
*/
Scseuil = Sc0^τ

/*
```

```
Recherche du motif dans la protéine de longueur m

AAj l'acide aminé j dans la fenêtre de la protéine
AAautj les acides aminés autorisés à la position j du motif
*/
Pour i=1 à m Faire                  /* boucle 1 */
      Pour j=1 à n Faire            /* boucle 2 */
            Si AAj+i-1 appartient à AAautj
                  Scj = Scj-1
            Sinon
                  Scj = Scj-1 / Pj

            Si Scj > Scseuil
                  Sortir boucle 2

      Si Scn <= Scseuil
            Écrire segment [AAi , AAi+n] dans le fichier résultat
            Écrire le taux d'information biologique obtenu
                  log Scn / log Sc0
```

4.1.4 Implémentation : biolcp

Nous avons programmé cet algorithme en langage C et l'avons intégré à la bibliothèque *biolcp*. Les fonctions disponibles (Annexe E7) permettent de rechercher un motif dans une protéine (*ProtMotif*), de rechercher les occurrences de chaque motif de Prosite dans une séquence protéique (*Proscan*) ou de rechercher les occurrences d'un motif dans une banque de séquences (*Pattinprot*). Ces recherches ont pour critère de sélection (*searchcrit*) soit l'identité stricte, soit un nombre maximal de positions fausses (« mismatch »), soit un taux minimal de ressemblance tel qu'il est défini dans notre algorithme.

```
int ProtMotif (char* protein, char* searchcrit, char* pattern);
int Proscan (char* fn_prot, char* searchcrit, char* fn_prositedb, char*
fn_prositedoc);
int Pattinprot (char* fn_pattern, char* searchcrit, char* fn_protdb);
```

Nous avons également inclus dans la bibliothèque *biolcp* des fonctions pour extraire les informations d'un fichier résultat obtenu par *pattinprot*. Ces routines testent le format du fichier (*PATTINPROT_IsItAPattinprotResultFile*), extraient le nom de la banque de séquences parcourue (*PATTINPROT_FresultReadLibraryName*) et positionnent le pointeur de fichier d'une entrée à l'autre (*PATTINPROT_FresultGoToFirstProt* et *PATTINPROT_FresultGoToNextProt*) :

```
extern int PATTINPROT_IsItAPattinprotResultFile (char* fn_res);
extern void PATTINPROT_FresultReadLibraryName (FILE* fp_res, char* libname,
long maxlen);
extern int PATTINPROT_FresultGoToFirstProt (FILE* fp_pattinprot);
extern int PATTINPROT_FresultGoToNextProt (FILE* fp_pattinprot);
```

Une entrée type (Figure 14) est encadrée par deux lignes de 80 tirets '-'. La première ligne contient la ligne d'identification d'une entrée protéique dans un fichier de séquences au format Pearson/FASTA condensé. Ensuite les occurrences du motif dans la protéine sont répertoriées les unes sous les autres. Chacune débute par le mot « Site » suivi des positions de début et de fin du motif dans la protéine, et du taux de ressemblance du motif trouvé avec le motif recherché. La ligne suivante

contient le motif trouvé dans la protéine. Ce motif est encadré par un certain nombre des acides aminés qui le précèdent ou le suivent dans la séquence protéique. Ce qui s'avère très utile dans le cas d'un alignement multiple de toutes les occurrences du motif.. La distinction entre les trois groupes d'acides aminés est réalisée par l'intercalation d'un symbole '_'[54]. Les positions fausses du motif trouvé sont indiquées en lettres minuscules, alors que les positions concordantes sont en majuscule. De plus les positions fausses du motif recherché sont indiquées explicitement sur la ligne suivante. Leur numéro et leur description sont clairement indiqués afin de lever toutes ambiguités lorsque le motif recherché contient plusieurs positions de longueur variable[55].

```
----------------------------------------------------------------------
>sw|P11219|AGI_ORYSA LECTIN PRECURSOR (AGGLUTININ).
Site :    46-   68, Similarity 92%
   phnlc_CsQFGYCGLGRDYCGTGCQSgAC_cssqr
   Bad position(s): 2 '[LIVMFYATG]', 10 '[FYWLIVSTA]'
Site :    59-   83, Similarity 87%
   lgrdy_CGTGCQSGACcSSQRCGSQGGGATC_snnqc
   Bad position(s): 4 '[WL]'
Site :   126-  154, Similarity 92%
   angel_CpNNMCCSQWGYCGLGSEFCGNGCQSgAC_cpekr
   Bad position(s): 2 '[LIVMFYATG]', 10 '[FYWLIVSTA]'
----------------------------------------------------------------------
```

Figure 14 : Exemple d'une entrée d'un fichier résultat de recherche de motif protéique

4.1.5 Disponibilité : ProScan, PattInProt, MPSA et NPS@

Les fonctions de recherche de motif ont également été incorporées dans deux logiciels indépendants : ProScan et PattInProt. Ils proposent la recherche des occurrences de tous les motifs de Prosite dans une protéine (ProScan) et la recherche de toutes les séquences d'une banque de protéines contenant des occurrences d'un motif protéique (PattInProt). Ces deux logiciels sont proposés dans l'ensemble logiciel MPSA disponible sur notre site Web (URL 29). Les routines *Proscan* et *Pattinprot* sont également intégrées dans notre logiciel MPSA et dans notre serveur Web NPS@.

Le logiciel PattInProt propose une recherche des séquences de protéines contenant plusieurs motifs distincts, et leur enregistrement dans un seul fichier résultat. Cette recherche est réalisée par des invocations répétées[56] de PattInProt. Lors de la recherche du motif i, il faut spécifier le nom du fichier résultat obtenu à l'étape précédente (si i > 1) et le nom du fichier qui contiendra la sous-base des séquences trouvées lors de cette étape i. Ainsi lors de chaque étape le fichier résultat précédent est modifié : les séquences contenant les motifs 1 à i sont conservées et les séquences ne contenant pas le motif i sont supprimées du fichier résultat. Chaque étape i de la recherche globale s'effectue avec un critère de sélection (identité, nombre de « mismatch » ou taux d'information biologique) indépendant de ceux des étapes précédentes. C'est pourquoi il est plus efficace de rechercher en premier le motif avec le critère de sélection le plus discriminant. Ainsi les recherches suivantes se feront sur une sous-base plus réduite et seront plus efficace.

Il est à noter que la création de la sous-base des séquences contenant le motif peut être réalisée même si la recherche ne concerne qu'un seul motif. Cette sous-base pourra alors être soumise à des méthodes d'analyse de séquences de protéine comme par exemple l'alignement multiple et la prédiction des structures secondaires sur une famille de protéines.

[54] le symbole « underscore »
[55] indiquées par un intervalle entre parenthèses, *e.g.* « A(12,18) ».
[56] autant de fois que de motifs protéiques recherchés.

Un exemple du gain apporté par notre méthode est le suivant. Nous recherchons le motif PS00881 de PROSITE sur SWISS-PROT 37 contenant 78 852 séquences. Ce motif est la signature du site d'épissage des protéines (son identificateur est « PROTEIN_SPLICING »). Sa syntaxe est [DNEG]-X-[LIVFA]-[LIVMY]-[LVAST]-H-N-[STC]. La recherche est effectuée avec un taux minimal d'information biologique conservée égal à 80%. 985 sites sont trouvés dans 959 séquences. Ce taux de 80% correspond au maximum à 2 positions fausses. En effectuant la même recherche mais en utilisant comme critère de sélection « mismatch = 2 », 64 823 sites sont trouvés dans 36 675 séquences. De plus la majorité des motifs trouvés dans le second cas n'ont pas les 2 positions strictes conservées : « -H-N- », alors qu'elles sont conservées avec notre méthode.

Une collaboration est actuellement en cours entre notre laboratoire et l'Institut Suisse de Bioinformatique (SIB, URL 48) afin de valider PattInProt lors d'une utilisation à grande échelle : une recherche des occurrences de tous les motifs de Prosite dans les séquences de la banque SWISS-PROT afin de mettre à jour les annotations de ces deux banques de données.

4.2 La bibliothèque « biolcp »

Pour nos différentes réalisations nous avons utilisé des bibliothèques existantes comme Vibrant ou ReadSeq. Mais nous avons également implémenté des méthodes originales ou publiées et les avons regroupées dans une bibliothèque. L'objectif de cette bibliothèque, que nous avons appelée *biolcp*, est de mettre à la disposition de la communauté de bioinformaticiens les différentes implémentations de méthodes biologiques que nous avons réalisées.

Parmi celles-ci il y a notamment la méthode originale de recherche d'un motif protéique dont nous avons parlé ci-dessus et la partie client de l'environnement MPSAweb dont nous parlerons plus en détail dans le chapitre 4.4. Mais *biolcp* contient également d'autres fonctions utiles au bioinformaticien pour l'analyse de séquences de protéine.

4.2.1 Le format MPSA de fichiers de données biologiques

En informatique et *a fortiori* en bioinformatique, la pléthore de formats de fichier différents est un problème récurrent. En effet il existe une quinzaine de formats pour les fichiers de séquences[57], une demi-douzaine pour les fichiers d'alignement multiple, pour les prédictions de structure secondaire de protéine presque autant de formats qu'il existe de méthodes ; *etc.* De plus beaucoup de ces formats n'ont pas de balises explicites qui permettent de les reconnaître[58]. Il faut alors se repérer sur des mises en forme ou des mots particuliers du texte, avec tous les risques d'erreur qui en découlent.

Nous avons mis au point un format de fichier permettant de stocker des alignements multiples, des structures secondaires, des caractéristiques physico-chimiques et également le résultat d'une ou plusieurs recherches de motifs protéiques suivant notre algorithme. Le format MPSA utilise une ligne bannière qui indique la nature et la taille des données du fichier. Ainsi une fois analysée la bannière et avant toute lecture des données, la mémoire nécessaire à leur stockage est allouée dynamiquement à l'aide de pointeurs adéquats et des routines standards du langage C. De plus il est possible d'insérer avant la ligne bannière des informations relatives à l'origine des données sans que cela ait des

[57] Dont une demi-douzaine pour les séquences protéiques.
[58] Comme le fait par exemple Clustal W en commençant la 1re ligne d'un fichier d'alignement par « CLUSTALW ».

conséquences sur la reconnaissance du format et la lecture. Par exemple nous plaçons avant la bannière des informations lorsque le fichier est issu de calculs effectués sur notre serveur Web NPS@.

Ce format est utilisé dans MPSA, sur notre serveur Web NPS@ et également dans les outils client-serveur MPSAweb.

4.2.2 Fonctions *biolcp* d'analyse de recherche de similarité dans les banques de séquences

Les premières fonctions que nous évoquerons sont celles qui permettent de reconnaître et d'analyser un fichier résultat de BLAST ou de FASTA, et de manipuler une banque de séquences au format Pearson/FASTA condensé (Tableau 17).

Tableau 17 : Bibliothèque *biolcp*, fonctions d'analyse des fichiers BLAST et FASTA : *blastio.h* et *fastaio.h*

Fonction	Action
`BLASTIO_IsItABlastResultFile`	Teste si le fichier est un fichier résultat de BLAST
`BLASTIO_FresultReadLibraryName`	Extrait le nom de la bibliothèque d'un fichier résultat de BLAST
`BLASTIO_FresultGoToSeqNamesBlock`	Positionne le pointeur de fichier au début du bloc des séquences dans un fichier résultat BLAST
`FASTAIO_AllocFastaIOSeq`	Alloue la variable utilisée pour manipuler les données FASTA
`FASTAIO_IsItACompactFastaDB`	Teste si le fichier est un fichier de séquences au format Pearson/FASTA condensé (dit « double pipe »)
`FASTAIO_IsItAFastaResultFile`	Teste si le fichier est un fichier résultat de FASTA
`FASTAIO_GoToNextEntry`	Positionne le pointeur de fichier au début de la prochaine entrée dans un fichier de séquences

`FASTAIO_GetSeq`
`FASTAIO_FastReadSeq`
`FASTAIO_ReadProtId`
`FASTAIO_ReadProtIdAndComments`
	Lit une entrée d'un fichier de séquences

`FASTAIO_GetSeqFromId`	Lit la séquence correspondant à l'identificateur
`FASTAIO_FresultReadLibraryName`	Extrait le nom de la bibliothèque d'un fichier résultat de FASTA
`FASTAIO_FresultGoToSeqNamesBlock`	Positionne le pointeur de fichier au début du bloc des séquences dans un fichier résultat FASTA

4.2.3 Fonctions *biolcp* relatives aux structures secondaires

Tableau 18 : Bibliothèque *biolcp*, fonctions relatives aux structures secondaires : *dsspio.h, secpred.h* et *secondary.h*

Fonction	Action
`DSSPIO_AllocMainData`	Alloue la variable utilisée pour manipuler les données DSSP (P 112)
`DSSPIO_IsItADsspStream` `DSSPIO_IsItADsspFile`	Teste si le fichier est un fichier DSSP
`DSSPIO_ReadDataFromStream` `DSSPIO_ReadDataFromFile`	Lit les données dans un fichier DSSP

Fonction	Action
`SEC_MpsaGetSizeFromFline`	Extrait les dimensions de la prédiction à partir de la ligne courante
`SEC_Read`	Lit un fichier de structure secondaire au format MPSA, PHD, SIMPA 96 ou PREDATOR
`SEC_ReadMpsa` `SEC_ReadMpsaScoreFile`	Lit un fichier MPSA
`SEC_ReadPredator`	Lit un fichier PREDATOR
`SEC_ReadSIMPA96`	Lit un fichier SIMPA 96
`SEC_ReadPHD`	Lit un fichier PHD
`SEC_WriteMpsaConsensusFile`	Ecrit un fichier consensus au format MPSA
`Garnier`	Méthode GOR 1 de prédiction de structure secondaire
`Gibrat`	Méthode GOR 2 de prédiction de structure secondaire
`Levin`	Méthode Homologue de prédiction de structure secondaire

La bibliothèque *biolcp* propose également l'implémentation de fonctions permettant de manipuler des structures secondaires (Tableau 18). Il s'agit par exemple de la lecture de fichiers résultats dans différents formats : MPSA, PHD, SIMPA 96 et PREDATOR, mais aussi de la prédiction de la structure secondaire d'une séquence d'acides aminés suivant 3 méthodes : GOR 1, GOR 2 et la méthode homologue.

4.2.4 Fonctions *biolcp* relatives aux profils physico-chimiques

Tableau 19 : Bibliothèque *biolcp*, méthodes de prédictions de caractéristiques physico-chimiques

Caractéristiques physico-chimiques	Auteurs
Accessibilité au solvant	Janin (P 110)
Antigénicité	Welling *et al.* (P 185) Parker *et al.* (P 150)
Flexibilité	Karplus et Schulz (P 116)
Hydrophobie, hydrophilie	Kyte et Doolittle (P 120) Hopp et Woods (P 106)
Hélices transmembranaires	Rao et Argos (P 158)

La bibliothèque *biolcp* contient sept méthodes de profils physico-chimiques appliquées aux séquences de protéines (Tableau 19). Les fonctions disponibles permettent d'effectuer la prédiction à l'aide d'une de ces méthodes mais également de sauvegarder les résultats dans un fichier au format MPSA (Tableau 20). Les échelles utilisées par ces sept méthodes sont indiquées dans l'Annexe F.

Tableau 20 : Bibliothèque *biolcp*, fonctions relatives aux profils physico-chimiques : *physchem.h*

Fonction	Action
`PHYSCHEM_ShiftingSegmentMethod`	Prédiction de profils physico-chimique avec l'algorithme du « segment déplacé » : Janin (P 110), Hoop et Wood (P 106), Welling et al. (P 185), Kyte et Doolittle (P 120), Argos et Rao (P 158)

Fonction	Action
`PHYSCHEM_KarplusSchulzFlexibility`	Flexibilité des protéines d'après Karplus et Schulz (P 116)
`PHYSCHEM_ParkerAntigenicity`	Antigénicité d'après Parker et al. (P 150)
`PHYSCHEM_WriteMpsaProfileFile`	Ecriture d'un fichier contenant les prédictions de caractéristiques physico-chimiques

4.2.5 Fonctions utilitaires de *biolcp*

Plusieurs fonctions utilitaires sont disponibles dans la bibliothèque *biolcp* afin de tester l'existence ou les droits d'écriture de fichiers ou de répertoires, d'identifier un fichier au format MPSA (Tableau 21, *lcpio.h*). Il est également possible de manipuler un fichier d'aide en ligne et d'obtenir le consensus d'une chaine de caractères (*lcphelp.h* et *lcputil.h*). La méthode du consensus (P 68) est utilisée dans MPSA sur les colonnes d'un alignement multiple mais également pour les consensus locaux et globaux de structures secondaires.

Tableau 21 : Bibliothèque *biolcp*, fonctions utilitaires : *lcpio.h*, *lcphelp.h* et *lcputil.h*

Fonction	Action

`FMAN_IsBinaryAvailable`
`FMAN_ExistingFile`
`FMAN_ExistingDir`
	Teste l'existence d'un fichier

`FMAN_WritableFile` `FMAN_WritableDir`	Teste les droits d'écriture
`FMAN_FilenameFromPath` `FMAN_DirnameFromPath`	Extrait le nom de fichier et le répertoire d'un chemin de fichier
`FMAN_CopyFile`	Copie un fichier dans un autre
`FMAN_MpsaFileGetBannerLine`	Récupère la ligne bannière d'un fichier au format MPSA
`FMAN_MpsaFileSec` `FMAN_MpsaFileAli`	Teste le type du fichier au format MPSA
`HELP_OpenHelpFile` `HELP_IsHelpFile`	Teste et ouvre le fichier uniquement si c'est un fichier d'aide
`HELP_IsChapterTitleLine`	Teste si la ligne courante est un titre de chapitre
`HELP_GoToChapter`	Positionne le pointeur de fichier sur le chapitre numéro N
`HELP_ReadChapterTitle`	Extrait les titres de chapitre d'un fichier d'aide
`GetConsensus`	Retourne le consensus d'une chaine de caractère (*e.g.* des acides aminés)

4.2.6 Mise à disposition

La bibliothèque est mise à la disposition de la communauté bioinformatique à travers une page Web (http://www.ibcp.fr/biolcp). Cette page propose le téléchargement de la bibliothèque et des fichiers d'en-tête. La bibliothèque est disponible compilée pour différentes architectures : stations de travail sous UNIX (IRIX 5.3 et IRIX 6.x, AIX 4.1, Solaris 5.6), mais également PC et Macintosh.

4.3 NPS@ : serveur Web d'analyse de séquences de protéine

Le serveur Web NPS@ (http://pbil.ibcp.fr) est constitué du serveur W3 Apache 1.3[59] interfacé à de nombreuses méthodes d'analyse de séquences de protéine. NPS@ est intégré depuis janvier 1998 au Pôle Bioinformatique Lyonnais (PBIL, URL 35).

Nos développements ont consisté à réaliser les interfaces en langage Perl entre le serveur Web Apache et les méthodes d'analyse. Ces interfaces sont des *cgi-bin* activés automatiquement par le biologiste lors de la soumission d'un formulaire d'analyse depuis l'une des nombreuses pages de paramétrage.

4.3.1 Position de NPS@ dans Internet

Considérons tout d'abord la position dans Internet de la machine qui héberge la partie NPS@ du PBIL. Cette station est une Silicon Graphics® Origin 200® TwinTower quadriprocesseurs. Cette machine a pour nom d'hôte 'bmw' et est connectée au domaine 'ibcp.fr'. Le réseau de l'IBCP est un réseau Starlan, c'est-à-dire un réseau en arbre actif avec la technique d'accès Éthernet. Le domaine 'ibcp.fr' est un sous-réseau du LAN de l'École Normale Supérieure de Lyon (domaine 'ens-lyon.fr'). Ce LAN est lui-même connecté au MAN de la recherche en région Rhône-Alpes : « Aramis », qui est à son tour interconnecté aux autres réseaux régionaux par le RNI[60]. Et l'ensemble de ces réseaux constitue RENATER, le WAN de la recherche en France. À son tour RENATER est connecté aux autres WAN par le NTI[61].

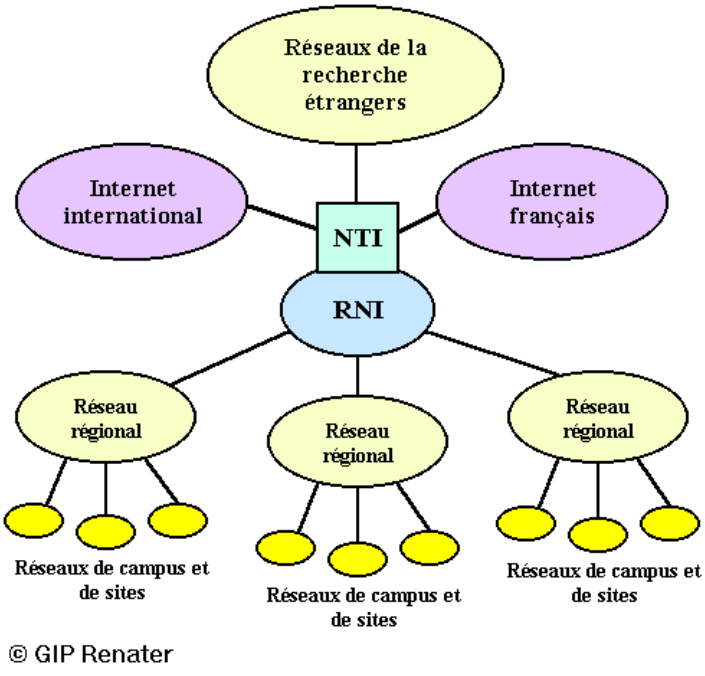


Figure 15 : Architecture de RENATER et sa connection avec les autres réseaux.

[59] http://www.apache.org
[60] RNI : Réseau National d'Interconnexion.
[61] NTI : Nœud de Transit International.

4.3.2 Organigramme du serveur NPS@

La page d'accueil de NPS@ (Figure 16) propose aux biologistes de nombreuses méthodes d'analyse de protéines :

- Recherche de similarité dans les banques de séquences : BLAST, FASTA et PattInProt,
- Ces méthodes sont utilisables sur plusieurs banques de séquences de protéine : SWISS-PROT, SP-TrEMBL, Non-redondante, Nrl3D,
- Méthodes d'alignement multiple : Clustal W et Multalin,
- De nombreuses méthodes de prédiction de structure secondaire de protéine : SOPM, SOPMA, méthode homologue version de 1986 (Levin) et de 1996 (SIMPA 96), méthode de double prédiction (DPM), méthode GOR version 1, 2 et 4, méthode MLR, méthode PREDATOR,
- Les méthodes de prédiction des caractéristiques physico-chimiques que nous avons évoquées à propos de *biolcp*,
- Une recherche des occurrences des motifs Prosite dans une protéine,
- Et de nombreux autres outils dont par exemple une prédiction des motifs structuraux « hélice-tour-hélice » ou de ceux dits « coiled-coil », une mise en évidence de certains acides aminés d'une séquence en les affichant en couleur dans la séquence.

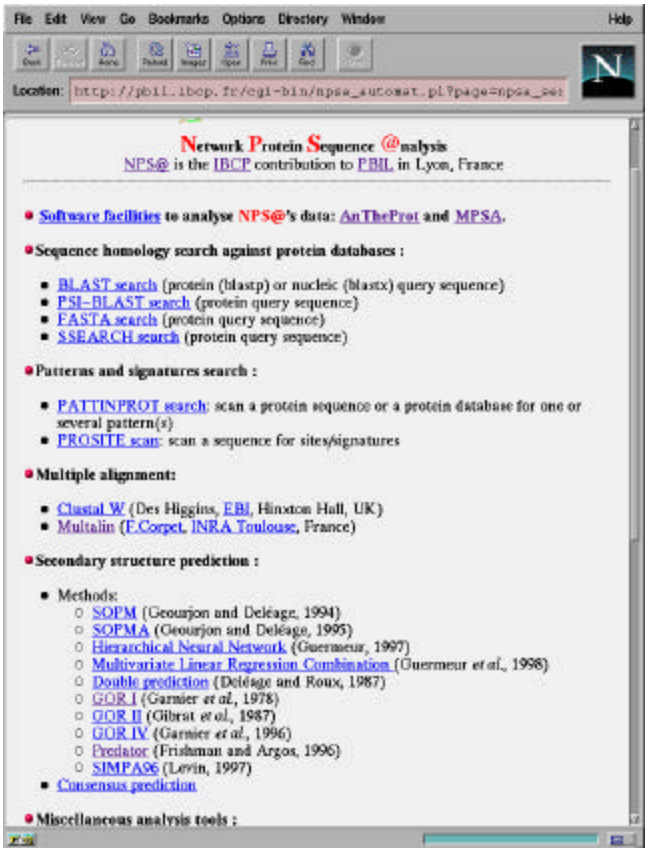


Figure 16: NPS@, la page d'accueil

Toutes ces méthodes se paramètrent au moyen de page Web où le biologiste n'a la plupart du temps qu'à entrer sa séquence[62] et parfois qu'à soumettre le formulaire. En effet les paramètres sont très souvent positionnés sur des valeurs optimisées. Le seul champ à remplir est alors celui de la séquence protéique. Et même ce champ est parfois déjà rempli lorsque la séquence a été obtenue à l'aide d'un autre outil de NPS@.

4.3.3 Interactivité des méthodes et des données

En effet toutes les méthodes disponibles sur NPS@ sont interconnectées. Par exemple les séquences obtenues lors d'une recherche de similarité avec BLAST, FASTA ou PattInProt peuvent être automatiquement alignées avec Clustal W ou Multalin. Ensuite, des prédictions de structure secondaire et leur consensus peuvent être incorporés à l'alignement multiple. Le biologiste n'a le plus souvent qu'à suivre les liens hypertextes ou cliquer sur les boutons de choix.

De la même manière de nombreux liens hypertextes sont proposés pour obtenir des détails supplémentaires sur certaines données à partir des sites Web originaux. Ces sites sont ceux des banques de séquences mondiales : SWISS-PROT, SP-TrEMBL, PDB, GenBank,…Mais ce sont également d'autres banques de données comme Medline, Prosite ou SCOP.

Afin de proposer un environnement complet entre tous nos outils, la majorité des résultats obtenus sur NPS@ peuvent être transférés à une session de MPSA sur la machine locale. Le biologiste clique sur le lien hypertexte intitulé « View data in MPSA ». Une session MPSA est ouverte et le fichier de données lui est transmis. Le format et le type des données sont automatiquement reconnus et MPSA affiche ces données avec les codages adéquats comme nous le verrons plus loin (*cf.* 4.5). Pour mettre en place ce processus, nous avons créé un sous-type MIME original : « application/x-mpsa », qui est utilisé pour le transfert du fichier de données entre le serveur W3 et le navigateur du biologiste.

4.3.4 Statistiques de NPS@

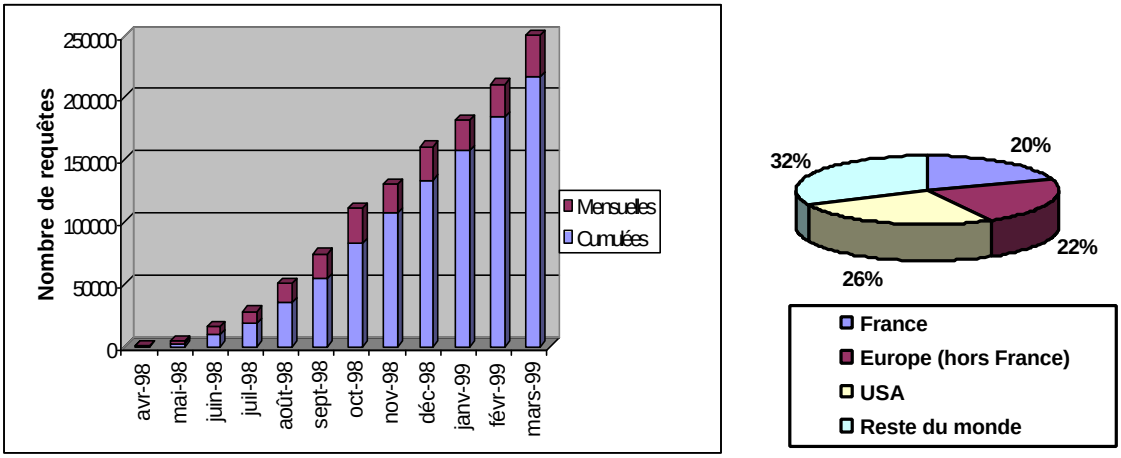


Figure 17 : NPS@, statistiques d'utilisation

Le serveur NPS@ traite en moyenne 1 000 requêtes par jour. Elles se répartissent de façon relativement équitable entre la France, le reste de l'Europe, les Etats-Unis et le reste du monde. Le site

[62] Ce qui peut être réalisé par un copier-coller depuis une autre application.

NPS@ est référencé par de nombreux sites Web internationaux (pour des exemples voir le Tableau 22).

Tableau 22: NPS@, quelques sites internationaux référençant notre serveur

Serveur	localisation	URL
InfoBioGen	GIS-InfoBioGen, Villejuif	http://www.infobiogen.fr
ExPASy et ses sites mirroirs	Swiss Institute of Bioinformatics, Genève Australian Proteome Analysis Facility, Sydney National Health Research Institutes, Taipei	http://www.expasy.ch http://expasy.proteome.org.au http://expasy.nhri.org.tw
Genome Web	Hinxton Genome Campus, UK	http://www.hgmp.mrc.ac.uk/GenomeWeb/prot-ed-format.html
Computational Protein Structure Group	Oak Ridge National Laboratory	http://grail.lsd.ornl.gov/protein_group/resource
CMS Molecular Biology Resource	San Diego Supercomputer Center	http://www.sdsc.edu/projects/ResTools/cmshp.html
Protein server	Harvard University	http://cyrah.med.harvard.edu/Proj/Bio/Protein/
NEE-HOW: the bioresource finder	Chinese Academy of Medical Sciences	http://moon2.cicams.ac.cn/wonderful/soft-online/protstru2.html
ABIM	Université Aix-Marseille 1	http://www-biol.univ-mrs.fr/english/logligne.html

4.4 MPSAweb : des outils client-serveur d'analyse de séquences de protéine

MPSA est capable de soumettre des requêtes à un serveur Web en utilisant le protocole HTTP (Figure 18). Or la plupart des serveurs Web proposant des analyses de séquence mettent en forme le résultat des méthodes, qui est toujours contenu dans de simples fichiers-texte. Pour cela ils extraient les informations essentielles et les présentent en utilisant les possibilités de mise en page proposées par le langage HTML (Chap. 1.1, Tableau 13).

4.4.1 Serveur MPSAweb

Ce n'est pas notre volonté de faire de MPSA un navigateur Web. C'est pourquoi MPSA n'est en aucun cas capable d'interpréter le langage HTML. La composante Web de MPSA n'est constituée que par la possibilité de soumettre des requêtes d'analyses biologiques et de recevoir et traiter les réponses. Ceci nécessite donc que les scripts CGI ou « cgi-bin »[63] du serveur concerné traitent le cas où la requête provient d'un client MPSA. Deux options sont possibles : soit modifier le script CGI en court-circuitant la mise en page des résultats afin de répondre par les fichiers bruts, soit installer des scripts CGI spécifiques à MPSA. Ces seconds scripts CGI sont plus simples puisqu'ils ne font qu'invoquer la méthode d'analyse désirée. La mise en page des résultats est en effet la fraction d'un script CGI la plus lourde à programmer.

[63] Petit exécutable accessible depuis une page Web *via* un lien hypertexte ou la soumission d'un formulaire. Ils sont généralement écrits avec le langage Perl.

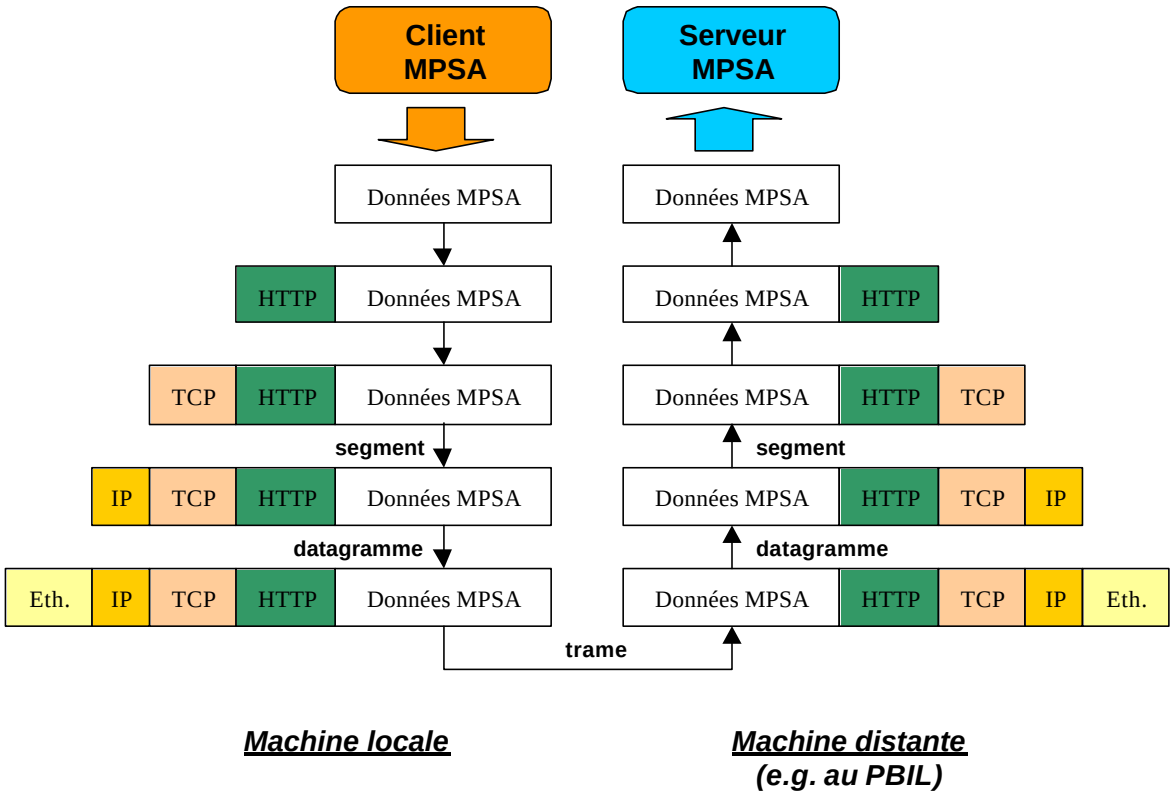


Figure 18 : MPSAweb : mise en forme des données lors d'une requête entre un client et un serveur MPSA

C'est pourquoi nous proposons un paquet de scripts CGI propres à MPSA permettant d'interfacer des méthodes de recherche de similarités dans les banques (BLAST, FASTA), d'alignement multiple (Clustal W, Multalin), de prédiction de structure secondaire. En effet MPSA est capable de manipuler ce type de données et d'en proposer une analyse graphique.

Tableau 23 : serveur MPSAweb, les différents script Perl et leur utilité

Script Perl	**Utilité**
mpsa_blast.pl, mpsa_fasta.pl	Recherche de similarités de séquence
mpsa_proscan.pl, mpsa_pattinprot.pl	Recherche d'occurrences de motifs protéiques
mpsa_clustal.pl, mpsa_multalin.pl	Alignement multiple de séquences
mpsa_db.pl	Banques de séquences : liste des banques disponibles, récupération de sous-ensembles.
mpsa_secmeth.pl, mpsa_secpred.pl	Prédiction de structure secondaire de protéine : liste des méthodes disponibles, application d'une méthode à une protéine.

4.4.2 Client MPSAweb : *mpsaweb.h*

Afin que ces fonctionnalités puissent être utilisées par d'autres programmeurs, nous avons intégré à la bibliothèque « biolcp » les routines en langage C pour émettre des requêtes de client MPSA (*cf. Annexe* E11). Ces routines constituent la partie client des outils client-serveur MPSAweb. Ainsi un programmeur intégrera très simplement à son logiciel la possibilité de soumettre des requêtes sur notre serveur Web au PBIL ou sur son serveur Web personnel. Il lui suffira alors d'installer sur son serveur Web le package de scripts Perl constituant la partie serveur-MPSA.

Tableau 24 : Les routines de client MPSAweb

Routine	**Action**
WWW_SubmitClustal	Requête d'alignement multiple avec Clustal W
WWW_SubmitMultalin	Requête d'alignement multiple avec Multalin
WWW_GetRemoteBankList	Récupération des banques de séquences disponibles sur le serveur distant
WWW_GetRemoteBankSubset	Récupération de séquences en passant comme argument leur identificateur
WWW_SubmitProScan	Requête Proscan
WWW_SubmitPattinprot	Requête Pattinprot
WWW_SubmitFasta	Requête FASTA
WWW_SubmitBlast	Requête BLAST
WWW_GetRemoteSecPredMethList	Récupération des méthodes de prédiction de structure secondaire de protéine disponibles sur le serveur distant
WWW_SubmitSecPred	Requête de prédiction de structure secondaire de protéine

4.5 MPSA : logiciel d'analyse de séquence de protéine

Le projet initial de MPSA débuta lors de mon stage de DEA de Biochimie en 1995. Plusieurs banques de données étaient déjà disponibles sur Internet. Les projets génomes se mettaient en place. Au vu de ces deux faits, il s'avérait que l'information biologique disponible allait croître très rapidement dans les années à venir. De plus le Web promettait une circulation et une publication des données et méthodes grandement simplifiées et accessibles à la grande majorité de la communauté scientifique.

4.5.1 Cahier des charges

Dans ce contexte et fort de l'expérience acquise dans le laboratoire avec le programme AnTheProt[64] (P 52, P 82, P 78, P 80), nous avons défini le projet d'un éditeur d'alignement multiple qui devait être une base solide à l'intégration future de techniques d'analyse de séquences de protéine. Le logiciel implémenté s'appela initialement MAE (Multiple Alignement Editor). Ce nom devenant trop restrictif avec l'intégration de nouvelles méthodes et algorithmes, le logiciel fut rebaptisé MPSA[65] (Multiple Protein Sequence Analysis). MPSA devait être une interface universelle, conviviale et efficace à l'analyse de séquence de protéine.

4.5.1.1 Hétérogénéité du parc informatique des laboratoires de biologie

Il est évident que le parc informatique utilisé par les biologistes est très disparate : il comprend en effet des ordinateurs individuels mais aussi stations de travail utilisant différents systèmes d'exploitation (UNIX, MacOS, Windows). Du point de vue de la conception de logiciel cela pose un gros problème de langage et d'option dans le code, ainsi que de conception de l'interface graphique

[64] AnTheProt : Analyze The Protein. http://www.ibcp.fr/ANTHEPROT/Documentation_antheprot.html
[65] MPSA : Multiple Protein Sequence Analysis. http://www.ibcp.fr/mpsa

suivant le système d'exploitation. Nous nous sommes affranchis de ces contraintes en utilisant le langage C norme ANSI et la bibliothèque graphique Vibrant du NCBI qui ont tous deux une « portabilité » élevée puisqu'ils fonctionnent sur toutes les architectures de machine. MPSA devait donc être universel pour le type de machine « supporté ».

4.5.1.2 Universalité des formats de données

MPSA devait aussi être universel du point de vue de l'interface aux méthodes intégrées. En effet l'utilisateur devait pouvoir soumettre des données à des méthodes d'analyse aussi diverses que l'alignement multiple de séquence, la recherche de similarité dans les banques de séquences, la recherche de motifs protéiques, la prédiction de structure secondaire, *etc.* et ceci de la manière la plus efficace et transparente possible. Ce qui implique que les paramètres doivent lui être accessibles par le biais de bouton poussoir, de menu déroulant, de choix entre plusieurs options, c'est-à-dire par des objets graphiques manipulables à l'aide de la souris.

L'utilisateur ne doit également pas avoir à se soucier du format des données en entrée et en sortie de chaque méthode ou algorithme. En effet le problème des formats des fichiers est un problème récurrent en informatique et ne manque pas de se poser aussi en bioinformatique. D'une méthode à l'autre et même à l'intérieur d'un même type de méthode, les formats varient le plus souvent du tout au tout, ce qui les rend complètement incompatibles les uns aux autres. Ce principe de formats gérés automatiquement devait aussi être appliqué au format des séquences d'acides aminés manipulées, qu'elles soient sous la forme de banques ou d'alignements multiples de séquences. En effet il existe une demi-douzaine de format de banque de séquences de protéine (EMBL, Pearson-FASTA, PIR,…) et autant de format d'alignement multiple (Clustal W, Multalin, MSF,…).

4.5.1.3 MPSA : un logiciel convivial et efficace

MPSA devait être convivial. Cela suppose que l'utilisateur ne doit pas avoir besoin de taper des lignes de commandes ou de faire des manipulations rébarbatives pour accéder à une fonctionnalité. Pour atteindre cet objectif, il était évident que l'interface graphique devait être simple, agréable à utiliser et reprendre tant que faire ce peut les concepts standardisés par les logiciels existant déjà (MacOS®, Windows®, Word®, *etc.*). Ceci afin qu'un utilisateur, découvrant le logiciel, trouve les fonctionnalités essentielles intuitivement, et ne soit obligé de recourir à la documentation que pour des fonctions évoluées.

MPSA devait être efficace. Des données de grande taille devaient être manipulables et analysables. Par exemple des alignements de plusieurs centaines de séquences de plusieurs milliers d'acides aminés devaient pouvoir être réalisés et optimisés. Les données, quelles qu'elles soient, devaient être affichées sous une forme très graphique utilisant des symboles et des couleurs reflètant une pré-analyse de ces données. Notamment l'utilisation de codage de couleur est très utile pour matérialiser une certaine information comme par exemple la conservation des acides aminés dans un alignement ou l'état conformationnel d'un acide aminé. En effet la couleur permet d'analyser beaucoup plus rapidemment les données, « d'un simple coup d'œil », et de mémoriser plus efficacement les informations ainsi obtenues.

4.5.2 Détails de l'implémentation

L'implémentation de MPSA est répartie sur 31 fichiers sources (35 240 lignes de codes) et 39 fichiers d'en-tête (5 942 lignes de codes). Elle comporte la définition de 846 fonctions. A lesquelles il

faut ajouter les 248 fonctions définies dans *biolcp*, ce qui représente 10 829 lignes dans les 20 fichiers sources et 3 481 lignes dans les 26 fichiers d'en-tête.

4.5.2.1 La variable « ali » : une liste chainée avec un accès incrémentiel

Un alignement[66] affiché dans la fenêtre principale est manipulé par MPSA grâce à une variable de type structure en langage C : la variable *Lcp_AliOrBseq* (*cf* Annexe C2). Cette variable regroupe toutes les données relatives à l'alignement dans différents champs : par exemple le nom de cet alignement (*name*), le chemin absolu de son fichier (*path*), un drapeau (*flg_need2besaved*) indiquant que l'utilisateur a modifié l'alignement et que par conséquent celui-ci devra être sauvegardé avant sa fermeture.

A côté de cette variable de l'alignement multiple, un ensemble de 4 variables contient les données relatives à ce que nous appelons les « structures » dans MPSA : les séquences d'acides aminés, les structures secondaires et les caractéristiques physico-chimiques. Cet ensemble de variables contient les données communes (*Lcp_Structure*) et spécifiques (*Lcp_PrimaryStructExtraData, Lcp_StructExtraDataSecondary, Lcp_StructExtraDataPhysChem*) à chaque type de données manipulées par MPSA. Ces structures sont organisées comme une liste linéaire chainée plusieurs fois et celle-ci est doublée d'un tableau de pointeurs sur chaque structure (Figure 19).

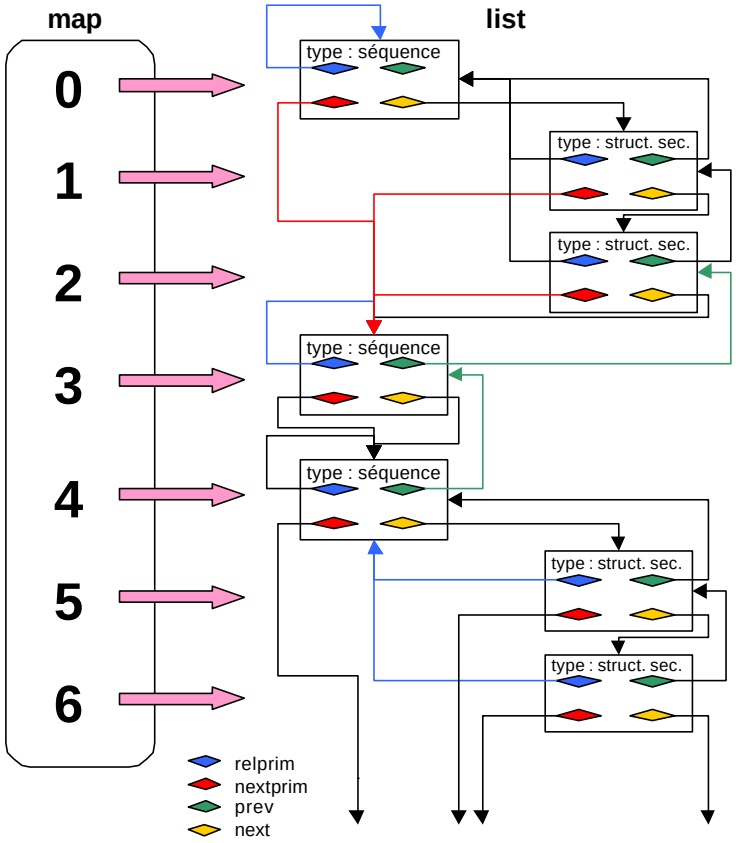


Figure 19 : Organisation des différentes variables décrivant un alignement dans MPSA

La liste chainée, pointée par le champ *list*, simplifie l'insertion et la suppression d'éléments. Ces opérations sont réalisées en modifiant les pointeurs *next* et *prev* de chaque élément encadrant

[66] Ou un ensemble de séquences lorsque celles-ci ne sont pas alignées.

l'élément inséré ou supprimé. Ces deux pointeurs référencent respectivement la structure suivante et la précédente. Chaque structure possède également deux autres pointeurs : *nextprim* et *relprim.* Le premier pointe directement sur la séquence suivante en ignorant les autres structures (secondaires et physico-chimiques) qui peuvent être intercalées. Le second pointe sur la séquence relative à la structure courante : par exemple dans le cas d'une structure secondaire il s'agit de la séquence primaire utilisée.

Le tableau, pointé par le champ *map*, permet un accès référencé à chaque structure et à chaque colonne de l'alignement. De cette manière l'alignement est manipulé comme un tableau à deux dimensions. Ceci apporte la rapidité nécessaire lors de l'affichage des séquences ou de la récupération des données relatives à une structure pointée par la souris. Car dans ces deux cas chaque séquence est repérée par son numéro dans l'enchainement des séquences.

4.5.2.2 Affichage d'une chaine de caractères en couleur

La bibliothèque Vibrant est relativement lente pour afficher du texte coloré (*cf.* 3.5.2). Nous avons remarqué qu'afficher M lignes de N caractères un par un avec P couleurs devient rapidement rédhibitoire lorsque N et P augmente : au-delà de 5-6 couleurs pour 10 lignes de 200 caractères. Or ce sont des valeurs courantes voire faibles pour l'affichage d'un bloc d'alignement multiple avec un codage de couleur significatif. C'est pourquoi nous utilisons l'algorithme suivant pour afficher une ligne de N acides aminés en utilisant P couleurs :

```
Procédure numcouleur(acide aminé)
Tableau de P lignes de N colonnes rempli par des espaces ‘ ‘

De i=0 à i=N-1
      Écrire l’acide aminé i à la colonne i dans la ligne numcouleur(acide
aminé i)

De i=0 à i=P-1
      Écrire la ligne i à l’écran dans la couleur i

De i=0 à i=N-1
      Écrire un espace ‘ ‘ à la colonne i dans la ligne numcouleur(acide
aminé i)
```

L'algorithme nécessite une procédure qui retourne le numéro de la couleur que prendra l' acide aminé courant à l'écran[67] et un tableau de P lignes de N colonnes initialisé une fois pour toutes[68] par des espaces.

La première étape consiste à écrire l'acide aminé courant dans la ligne correspondant à sa couleur. Elle permet de répartir tous les acides aminés de la ligne dans les P lignes correspondant aux P couleurs. Si toutes ces lignes sont superposées alors on obtient la séquence complète sans trou.

Ensuite ces P lignes des couleurs sont affichées à l'écran avec un seul appel de routine d'écriture de texte. Ce qui fait P appels d'une routine graphique au lieu des N appels si les acides aminés étaient écrits un par un. Une alternative pourrait être de n'effectuer qu'une seule invocation de routine graphique par bloc d'acides aminés consécutifs de la même couleur. Mais cet autre algorithme

[67] En fait ce n'est pas une procédure (relativement lente) qui est utilisée, mais une indirection (beaucoup plus rapide) sur les numéros des couleurs en fonction du code ASCII de l'acide aminé ou du numéro de la colonne.

[68] En fait uniquement au démarrage de MPSA et lors de redimensionnements de la fenêtre principale.

ne serait plus efficace que lorsque les portions conservées de l'alignement ne sont pas beaucoup fragmentées[69].

Pour terminer, les acides aminés stockés dans les P lignes sont remplacés par des espaces afin que le tableau soit prêt pour l'affichage de la ligne suivante. Car cet algorithme est répété M fois pour écrire les M séquences du bloc de l'alignement.

Cet algorithme est utilisé pour implémenter les fonctions d'affichage des lignes des différentes structures d'un alignement dans MPSA (*cf.* 4.5.2.1). En effet les routines d'affichages sont très souvent sollicitées et pour cette raison doivent être optimisées. C'est pourquoi plutôt que d'effectuer des tests coûteux en temps de calcul nous préférons utiliser un pointeur de fonction sur la bonne routine d'affichage : il s'agit du champ *drawline* de la variable *Lcp_Structure* (*cf.* Annexe C1).

4.5.2.3 Redimensionnement d'une fenêtre

Un autre inconvénient que nous avons évoqué précédemment est le fait que la bibliothèque Vibrant ne gère pas le redimensionnement d'une fenêtre. De plus, le seul type de fenêtre qui invoque une fonction de retour[70] lors d'une modification de taille est le type *DocumentWindow*. Nous utilisons ce type de fenêtre pour l'affichage des séquences (fenêtre principale) et pour les fenêtres affichant des graphes (par exemple le détail d'une prédiction de structure secondaire).

Dans ces cas la méthode employée est de supprimer les objets contenus dans la fenêtre redimensionnée. Il faut évidemment prendre soin de stocker les variables associées à ces objets. Puis nous reconstruisons tous ces objets en adaptant leur taille à la nouvelle taille de la fenêtre. Et enfin nous leur « ré-associons » les variables stockées auparavant.

Ci-dessous est indiquée la routine correspondante pour la gestion du redimensionnement de la fenêtre principale.

```
static void WINMAIN_Resize (WindoW win_main)
{
      int alipanwid=200, alipanhei=100;
      PoinT pt;
      RecT rect;
      Lcp_AliDisplay* disp=(Lcp_AliDisplay*)GetWindowExtra (win_main);
      prdev_nf (WINMAIN_Resize);

      if (disp->ali != NULL)
            DISPLAY_CalibrateSize (disp);

      /* panneau d'affichage de l'ali */
      UseWindow (win_main);
      ObjectRect (win_main, &rect);
      alipanwid = rect.right-rect.left-disp->lostwid;
      alipanhei= rect.bottom-rect.top-disp->losthei;
      GetPosition (disp->pan_ali, &rect);
      Remove (disp->pan_ali);
      pt.x = rect.left;
      pt.y = rect.top;
SetNextPosition (win_main, pt);
      disp->pan_ali = AutonomousPanel4 (win_main, alipanwid, alipanhei,
DrawProc
                  , SCROLL_Vertical, SCROLL_Horizontal, 0, NULL, NULL);
```

[69] Dans le cas d'un affichage de couleurs en fonction de cette conservation des acides aminés évidemment.
[70] Call-back

```
        disp->bar_h = GetSlateHScrollBar ((SlatE)disp->pan_ali);
        disp->bar_v = GetSlateVScrollBar ((SlatE)disp->pan_ali);
        SetPanelClick (disp->pan_ali, MOUSE_StructClick, MOUSE_StructDrag
                    , MOUSE_StructHold, MOUSE_StructRelease);
        if (DISPLAY_RegisterAliPanel (disp)) {
              WINMAIN_Close (win_main);
              QuitProgram ();
        }
        if (disp->ali != NULL) {
              UPDATE_MargeBelowAli (disp);
              DISPLAY_CalibrateSize (disp);
        }
        return;
}
```

4.5.3 L'interface graphique

L'interface graphique de MPSA se présente sous la forme de fenêtres indépendantes pouvant être classées suivant les fonctionnalités qu'elles proposent (Figure 20). Toutes les manipulations ou presque sont accessibles par la souris.

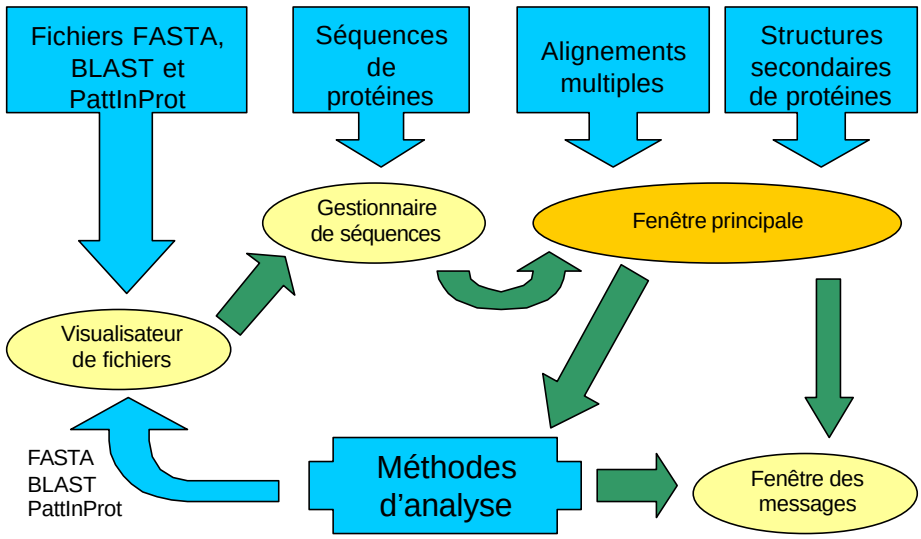


Figure 20 : MPSA, organigramme des fenêtres composant l'interface graphique.

4.5.3.1 Fonctionnalités liés à la souris

A part quelques données, comme les noms de fichiers, qui doivent être entrées au clavier, tous les outils, méthodes et fonctionnalités de MPSA sont accessibles par la souris. Toutes les actions disponibles dans les menus, les paramètres de toutes les méthodes, l'impression, les sélections et déplacements des séquences et structures associées ; toutes ces actions sont réalisées à l'aide de la souris.

Et notamment, diverses informations sont disponibles lorsqu'un alignement est affiché dans la fenêtre principale. Un clic de souris sur un acide aminé d'une séquence affiche différents paramètres de cet acide aminé dans la fenêtre des messages : le nom et le numéro de la séquence à laquelle il appartient, sa position dans cette séquence (*i.e.* position sans gap), sa position dans l'alignement (*i.e.* position avec gaps). Les mêmes types de paramètres sont également affichés lorsque l'élément pointé appartient à une structure autre qu'une séquence.

De la même manière un clic sur un identificateur de séquence retourne les informations disponibles sur cette séquence. Et un double-clic sur un identificateur provoque l'affichage du fichier contenant la séquence correspondante ou le détail de la prédiction lorsqu'il s'agit d'une structure secondaire ou de caractéristiques physico-chimiques.

4.5.3.2 La fenêtre principale

La fenêtre principale de MPSA est celle qui affiche les séquences d'acides aminés. Elle est le pivot de l'application. Elle est en permanence présente dans l'environnement graphique car elle permet d'accéder à toutes les fonctionnalités essentielles de MPSA.

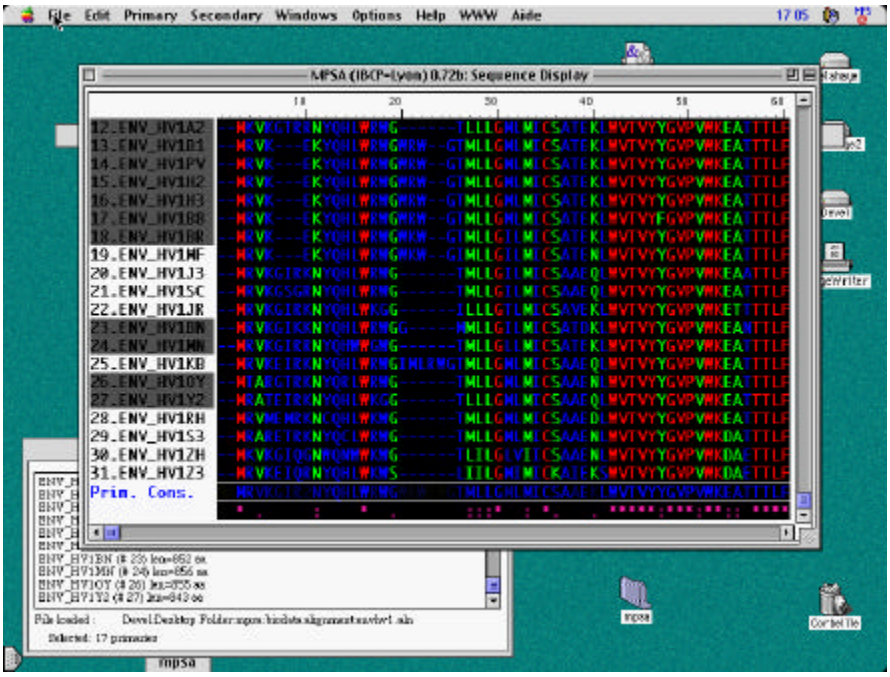


Figure 21 : Fenêtre principale de MPSA.

4.5.3.2.1 Les menus de la fenêtre principale

Ce sont les menus principaux de MPSA. Ils contiennent les accès à toutes les méthodes et outils disponibles dans ce logiciel. Les différentes fonctionnalités sont réparties dans ces menus suivant leur champ d'application (Tableau 25). Tous ces menus et leurs options se manipulent avec la souris. De plus beaucoup de ces options peuvent également être atteintes par un raccourci-clavier, ce qui augmente très souvent l'efficacité de la manipulation des données.

Tableau 25 : Les menus principaux de MPSA

Intitulé du menu	**Champ d'application**
File	Gestion de fichiers
Edit	Édition des données
Primary	Fonctionnalités propres aux séquences d'acides aminés (structure primaire)
Secondary	Fonctionnalités propres aux structures secondaires
Windows	Gestion des fenêtres de MPSA
Options	Personnalisation de MPSA

Intitulé du menu	Champ d'application
Help	Rubriques d'aide
WWW	Fonctionnalités propres au mode connecté en tant que client Web

4.5.3.2.2 La zone d'affichage des séquences

Cette fenêtre comporte une zone d'affichage des séquences d'acides aminés que le biologiste désire étudier. La zone d'affichage des séquences comporte plusieurs parties (Figure 21).

Tout d'abord la portion centrale contient l'affichage des acides aminés réalisé en utilisant leur code IUPAC à une lettre. Le sens de lecture est le sens habituel : l'extrémité N-terminale de la séquence est à gauche et l'extrémité C-terminale à droite de l'écran. Les acides aminés de ces séquences peuvent être alignés ou non, suivant qu'elles fassent partie d'un alignement multiple ou pas. Ensuite, les noms des protéines sont indiqués à gauche, chacune sur la même ligne que sa séquence d'acides aminés. Notons au passage que peuvent également être affichées dans cette zone des prédictions de structure secondaire et de caractéristique physico-chimique, mais nous y reviendrons plus en détail dans les chapitres correspondants (*cf.* 4.5.5.4 et 4.5.5.6).

Une échelle de repérage est affichée au dessus des séquences. Les dizaines sont matérialisées par un trait vertical surmonté de la valeur courante et chaque autre colonne par un point. Et en-dessous des séquences une séquence consensus et un indicateur de la conservation des acides aminés sont affichés. Ces deux outils ne sont évidemment affichés que dans le cadre d'un alignement multiple. Ce sont de bons indicateurs de la qualité de l'alignement multiple affiché.

La séquence consensus indique pour chaque position l'acide aminé représenté le plus grand nombre de fois dans la colonne. De plus cette séquence consensus est affichée avec un codage de couleurs particulier qui apporte une autre information : la fréquence de cet acide aminé dans cette colonne, de 0 à 100%. Par exemple sur la Figure 21, la séquence consensus est affichée avec un dégradé de couleur du bleu foncé (faible fréquence de cet acide aminé) à bleu clair (forte fréquence)

La ligne en-dessous de cette séquence consensus indique la conservation des acides aminés pour chaque position. Des symboles sont utilisés pour matérialiser les trois états : conservation d'un acide aminé, présence d'acides aminés similaires, acide aminé muté fréquemment. Le symbole '*' indique la conservation d'un acide aminé à cette position de l'alignement, c'est-à-dire dans cette colonne. Les symboles '.' et ':' indiquent que les acides aminés présents dans cette colonne appartiennent respectivement à un groupe de faible ou forte similarité. Les groupes de similarité utilisés sont ceux définis dans la méthode d'alignement multiple Clustal W (P 180).

4.5.3.2.3 Les ascenceurs pour parcourir les séquences

Lorsque les séquences sont en nombre trop important ou d'une taille trop grande pour tenir à l'écran, deux possibilités sont offertes. La première consiste à agrandir la fenêtre à l'aide de la souris et réduire la taille de la police de caractères utilisée pour afficher les séquences (*cf.* 4.5.5.8). Mais cette solution est vite limitée par la taille de l'écran et la mauvaise lisibilité d'une police trop petite. La seconde consiste à utiliser les ascenseurs disposés à droite et en bas de la zone d'affichage des séquences (Figure 21). Ces ascenseurs permettent de faire défiler les séquences respectivement verticalement et horizontalement. Le biologiste est alors en mesure d'afficher la portion d'alignement qu'il désire. Les deux ascenseurs fonctionnent classiquement en mode pas à pas (séquence par séquence verticalement et acide aminé par acide aminé horizontalement) ou en mode page par page.

4.5.3.3 La fenêtre des messages

La deuxième fenêtre apparaissant au début d'une session de MPSA est la fenêtre des messages. Elle comporte dans sa partie inférieure deux champs (Figure 22) où sont indiqués à l'instant courant le nom du fichier de séquences affiché (alignement ou banque de séquences) et le nombre de structures primaires et secondaires sélectionnées par le biologiste.

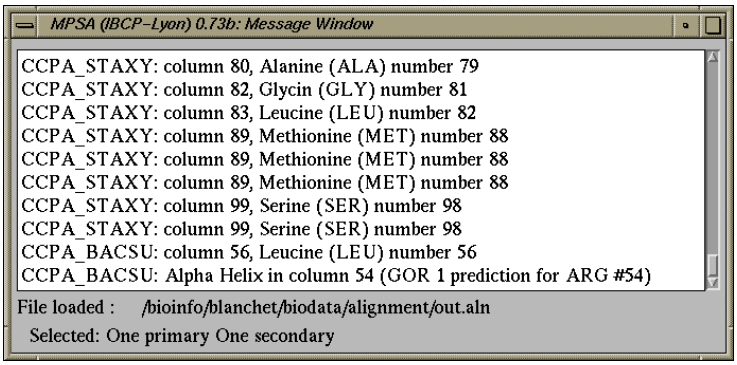


Figure 22 : MPSA, la fenêtre des messages

Au dessus, un champ de taille plus importante fait office de tableau déroulant où sont affichés tous les messages de MPSA pour l'utilisateur. Ces messages peuvent être des messages d'erreurs à la suite d'une mauvaise manipulation de l'utilisateur ou de l'échec dans l'exécution d'une tâche. Ce peuvent être aussi des informations que l'utilisateur a demandées : des informations sur l'acide aminé pointé par la souris, les informations disponibles sur une séquence ou sur une structure secondaire, *etc.*

4.5.3.4 Le gestionnaire de séquences

Un outil essentiel de MPSA est le gestionnaire de séquences. Cette fenêtre (Figure 23) est un point névralgique de la gestion des séquences non-alignées dans MPSA. En effet lors de l'ouverture de fichiers contenant des séquences de protéine, MPSA en extrait les identificateurs et les affiche dans ce gestionnaire de séquences.

Il comporte un champ de grande taille (Figure 23) où sont affichées des informations sur chaque séquence : leur identificateur (le cas de la Figure 23), leur longueur, les commentaires disponibles à leur propos et le format du fichier duquel elles ont été extraites. À droite, des boutons poussoirs permettent d'effectuer différentes actions : sélectionner toutes les séquences, annuler la sélection, supprimer les séquences sélectionnées, fermer la fenêtre ou extraire les séquences sélectionnées. L'extraction des séquences signifie extraire du fichier les séquences d'acides aminés et les afficher dans la fenêtre principale.

Il est important de noter que cette extraction se fait d'après le nom de fichier associé à l'identificateur de séquence. Parfois ce nom de fichier n'est plus disponible. C'est le cas lorsque les identificateurs proviennent d'un fichier-résultat des méthodes BLAST, FASTA ou PattInProt. Si la méthode a été utilisée sur une autre machine[71], alors la banque contenant les séquences similaires n'est pas forcément disponible sur la machine locale. Dans ce cas MPSA demande à l'utilisateur de lui indiquer un serveur Web où aller chercher les séquences en question. Et si la réponse du serveur est négative alors le biologiste peut indiquer un autre serveur jusqu'à trouver les séquences, ou décider de suspendre la recherche.

[71] Cas d'une requête sur un serveur Web distant.

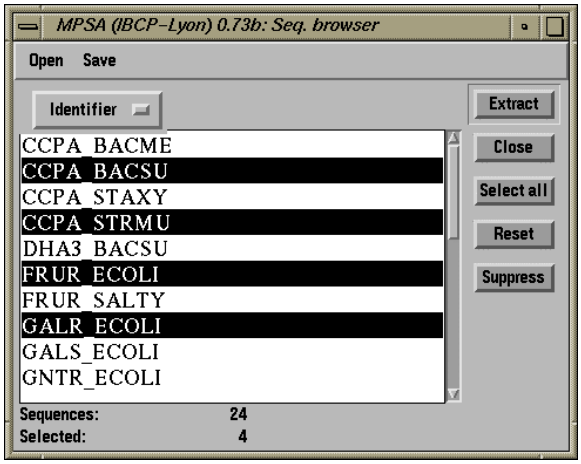


Figure 23 : MPSA, le gestionnaire de séquences

4.5.3.5 Le visualisateur de fichier

Le visualisateur de fichier est une fenêtre de MPSA qui permet d'afficher un fichier-texte quel que soit son format. Le fichier est simplement accessible en lecture, et il peut être imprimé. C'est utile lorsque le fichier considéré est dans un format qui n'est pas reconnu par MPSA.

Cet utilitaire est aussi employé pour l'affichage des fichiers-résultat de certaines méthodes d'analyse telles que FASTA, BLAST ou PattInProt. Dans ce cas une nouvelle fonctionnalité est disponible dans le menu « file » de cette fenêtre. Il s'agit de l'option « Extract seq. Id. ». Utiliser cette option permet, comme son nom l'indique, d'extraire tous les identificateurs de séquence contenus dans le fichier. C'est-à-dire extraire tous les identificateurs des protéines similaires à la séquence recherchée avec une des méthodes évoquées au dessus. Et les identificateurs extraits sont transférés au gestionnaire de séquences (Chap. 4.5.3.4). C'est ainsi un moyen commode de récupérer les protéines d'une même famille pour les inclure au processus d'analyse.

4.5.3.6 Les fenêtres d'affichage de graphes

Lors d'une analyse de protéines, le biologiste peut désirer visualiser la qualité de la prédiction d'une structure secondaire ou les zones les plus conservées de l'alignement. L'affichage de ces informations s'effectue dans une fenêtre d'un type particulier que nous pouvons nommer fenêtre de graphe[72]. Elle est composée d'une zone principale où s'affichent les graphes et de zones accessoires en haut et/ou en bas où s'affichent des informations relatives aux graphes (Figure 24).

Les graphes affichés disposent toujours d'un curseur vertical que l'utilisateur peut déplacer à l'aide de la souris[73]. À chaque déplacement, des informations liées à la position du curseur sont mises à jour dans la zone au dessus des graphes. Ces données peuvent être par exemple l'acide aminé correspondant, sa position dans la séquence, les scores de la prédiction de chaque état conformationnel pour cet acide aminé dans le cadre d'une prédiction de structure secondaire.

[72] Même si parfois dans le cadre de certaines méthodes, ce ne sont plus des graphes mais d'autres objets graphiques qui sont affichés.
[73] Soit par simple clic à une position donnée, soit par un « drag » sur toute une portion des graphes.

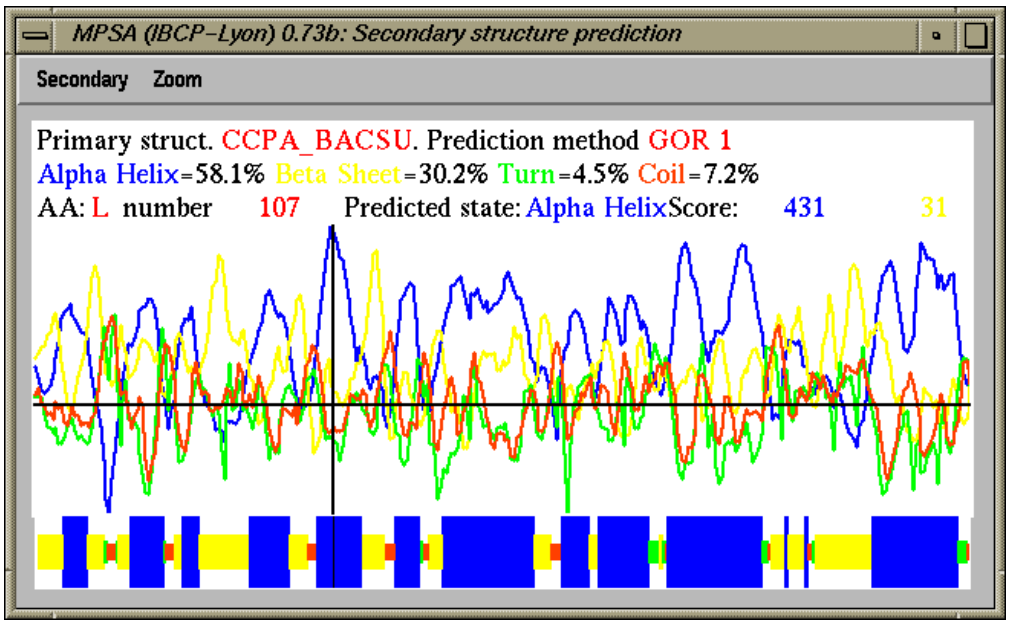


Figure 24 : MPSA, une fenêtre de graphe typique : la vue de détail d'une prédiction de structure secondaire.

Si l'échelle des abscisses est trop longue pour tenir à l'écran, c'est le cas lorsqu'une séquence est très longue, certains points ne seront pas affichés à l'écran. Deux pixels contigus auront alors un incrément de la valeur des abscisses supérieur à un. L'utilisateur peut n'afficher qu'une portion des graphes afin de grossir certains détails. Il lui suffit de délimiter la zone à agrandir en positionnant un deuxième curseur[74] et en utilisant le menu « zoom » de la fenêtre.

Une fonctionnalité essentielle de ces graphes est le couplage du curseur avec l'alignement affiché dans la fenêtre principale de MPSA. En effet lors du déplacement du curseur, l'alignement multiple subit le même déplacement. Ainsi les données des graphes affichées par le déplacement du curseur concernent la portion visible de l'alignement. Il est alors aisé de relier ces données à l'alignement.

4.5.4 Deux modes d'utilisation : non-connecté et client Web

L'analyse de séquence nécessite d'une part une très grande interactivité et des temps de réponse rapides de l'application utilisée et d'autre part l'utilisation de banques de données à jour et des calculs complexes.

Afin de proposer une solution à ce paradoxe, MPSA propose un mode de fonctionnement très efficace: le biologiste peut à l'aide de MPSA et de ses nombreux outils effectuer des analyses de séquences de protéine localement et lorsque il le désire, utiliser des méthodes disponibles sur des serveurs Web distants[75].

4.5.4.1 Choisir le serveur Web à interroger

Cette faculté de soumettre des requêtes à un serveur Web est très récente dans MPSA. C'est pourquoi ce choix est pour l'instant limité au Pôle Bioinformatique Lyonnais.

[74] En maintenant la touche « shift » enfoncée lors d'un clic ou d'un drag de la souris.

[75] Comme nous le proposons au Pôle BioInformatique Lyonnais (PBIL) pour des méthodes très lourdes à mettre en œuvre comme BLAST ou SOPMA.

Mais nous envisageons d'agrandir ce choix à l'aide de collaborations avec des serveurs proposant des méthodes très pointues disponibles uniquement à cet endroit, ou avec des serveurs généralistes répartis sur le globe afin de proposer à l'utilisateur de MPSA un temps de réponse toujours efficace (Chap. 4.4). La liste des serveurs compatibles MPSA est maintenue sur le site Web de MPSA (URL 29).

4.5.4.2 Connaître les possibilités d'un serveur Web

Afin de permettre une certaine souplesse et une certaine évolutivité à ce système client-serveur. MPSA interroge chaque nouveau serveur Web utilisé pour connaître ses fonctionnalités. Cette interrogation porte d'une part sur les banques de séquences de protéines disponibles et d'autre part sur les méthodes de prédiction de structure secondaire proposées (Figure 25). De cette manière toute nouvelle banque de séquences ou toute nouvelle méthode de prédiction mise en place sur un serveur connu sera accessible par MPSA sans que le biologiste est besoin de rapatrier une nouvelle version de MPSA. Et tout nouveau serveur Web utilisé indique à MPSA quelles sont les données et méthodes qu'il propose.

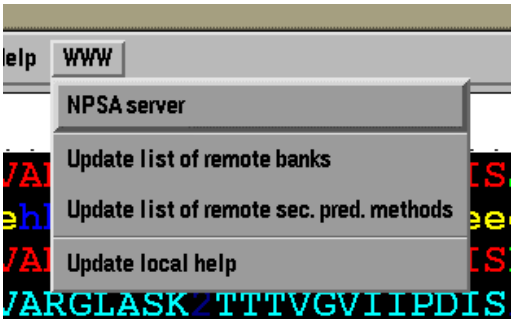


Figure 25 : Obtention de la liste des fonctionnalités proposées par un serveur Web compatible MPSA : le menu WWW de MPSA.

4.5.5 Les différentes fonctionnalités

Nous allons maintenant énumérer les différents outils et méthodes proposés par MPSA au biologiste afin de l'aider dans son analyse de séquences de protéine. Nous utiliserons pour cela l'ordre « classique » des étapes de l'analyse de la séquence d'une protéine inconnue : rechercher des séquences similaires à celle-ci, les aligner afin de travailler sur la famille de protéines, analyser et modifier l'alignement si besoin est.

4.5.5.1 Les formats disponibles en entrée et en sortie

La premiére étape de l'analyse d'une ou plusieurs séquences protéiques est de « charger » ces séquences dans MPSA. Comme nous l'avons vu précédemment (Chap. 4.3.1), les formats de données biologiques sont nombreux et variés. Il est pénible à un utilisateur de passer de l'un à l'autre lorsqu'il veut soumettre à une méthode les résultats d'une autre [76], ou de se demander dans quel format est le fichier de données qu'il souhaite analyser.

C'est pourquoi les procédures de gestion des fichiers on été conçues dans MPSA avec le souci d'alléger le biologiste des tracas liés au format des fichiers de données. Ainsi MPSA reconnaît

[76] D'où l'intérêt d'intégrer plusieurs méthodes dans un seul logiciel qui se charge de transférer les données de l'un à l'autre.

automatiquement un grand nombre de formats de fichier parmi les plus utilisés par la communauté scientifique (**Erreur! Référence non valide pour un signet.**). L'ouverture d'un fichier se fait à l'aide du menu « File » et de l'option « Open »[77], MPSA reconnaît automatiquement le format du fichier et traite les données en conséquence.

Tableau 26 : Formats de fichier reconnus par MPSA.

Nom du format	**Type des données**
EMBL	Séquence
Pearson/FASTA	Séquence
PIR	Séquence de protéine
Clustal W	Alignement multiple
Multalin (MSF)	Alignement multiple
PHD	Structure secondaire de protéine
SOPMA	Structure secondaire de protéine
DSSP	Structure secondaire de protéine
FASTA	Recherche de similarité de séquences
BLAST	Recherche de similarité de séquences
MPSA	Divers (alignement multiple, structure secondaire, caractéristiques physico-chimiques,…)

De la même manière MPSA propose de nombreux formats en sortie afin de pouvoir transférer ces données à d'autres outils informatiques. Notamment tous les formats de séquence disponibles en lecture le sont aussi en écriture, le format Clustal W est disponible pour les alignements, et évidemment le format MPSA. De plus, deux formats de mise en page sont disponibles : Adobe® PostScript et Microsoft® RTF. Ces formats sont accessibles à travers les options standards du menu « File » (Figure 26).

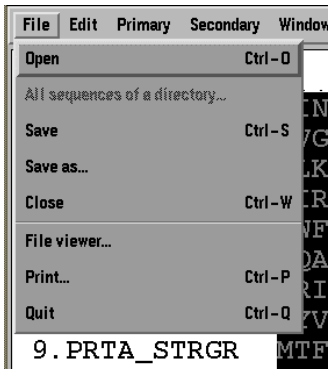


Figure 26 : MPSA, le me nu de gestion des fichiers

4.5.5.2 Recherche de similarité

Une analyse de séquence débute souvent par la recherche des séquences similaires déjà répertoriées dans les banques de séquences. En effet nous avons vu qu'il était plus fiable de travailler

[77] les accès aux options des menus seront par la suite indiquées comme suit « menu➔sous-menu➔…➔option ».

sur une famille de protéines plutôt que sur une séquence isolée (Chap. 2.3.3). De plus si la similarité entre les séquences est suffisamment bonne, les données des séquences connues peuvent être extrapolées à la séquence inconnue.

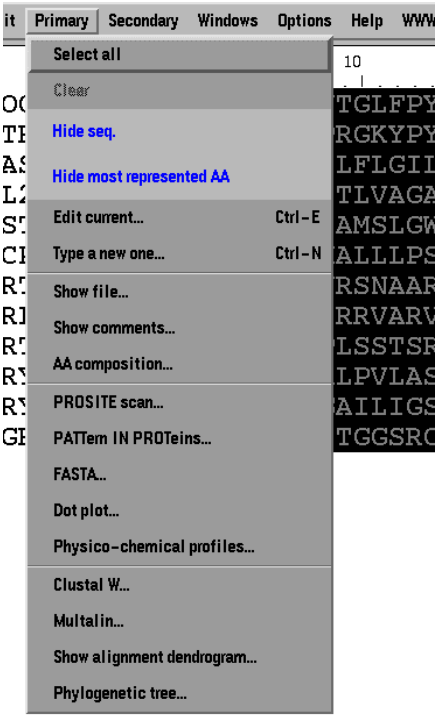


Figure 27 : MPSA, menu contenant les fonctionnalités relatives aux séquences d'acides aminés

MPSA propose plusieurs techniques pour identifier ces séquences similaires. Les recherches peuvent être effectuées à l'aide de la séquence complète ou d'un fragment de cette séquence en utilisant les méthodes heuristiques FASTA et BLAST d'alignement par paire (Chap. 2.3.2). Elles peuvent aussi utiliser des techniques de recherche d'un motif de type PROSITE suivant une méthode que nous avons mise au point :PattInProt (Chap. 1.1). Toutes ces méthodes sont accessibles par le menu « primary » de MPSA (Figure 27). La méthode utilisée à travers ce menu est appliquée à toutes les séquences sélectionnées dans la fenêtre principale.

4.5.5.2.1 Recherche à l'aide d'une séquence

Les deux méthodes proposées par MPSA pour rechercher une séquence dans les banques sont FASTA et BLAST. Les paramètres par défaut de ces méthodes sont utilisés lors du calcul. Le biologiste choisit la banque de séquence sur laquelle portera la recherche (Figure 28). Il choisit également le répertoire dans lequel seront enregistrés les fichiers résultats. Le biologiste peut ainsi sauvegarder les résultats de ses recherches en fonction du sujet dans des répertoires différents.

Les deux méthodes sont également disponibles pour un calcul sur un serveur Web distant. Le biologiste choisit le serveur Web auquel soumettre la recherche et choisit la banque de séquences parmi celles qui sont disponibles sur ce serveur.

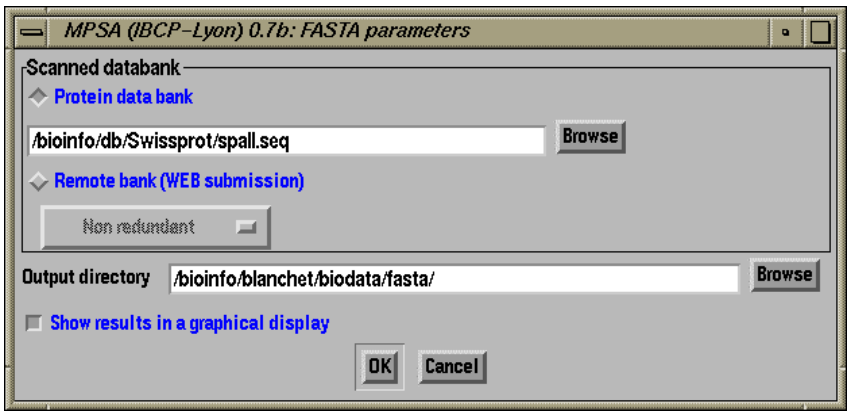


Figure 28 : MPSA, fenêtre de paramétrage de FASTA.

Le fichier-résultat généré par la méthode FASTA ou BLAST est automatiquement affiché lorsque les calculs sont terminés. Cet affichage est réalisé dans le visualisateur de fichier (Figure 29). Le biologiste peut alors analyser le fichier-résultat et en extraire les séquences similaires (*cf.* 4.5.3.5 : Le visualisateur de fichier).

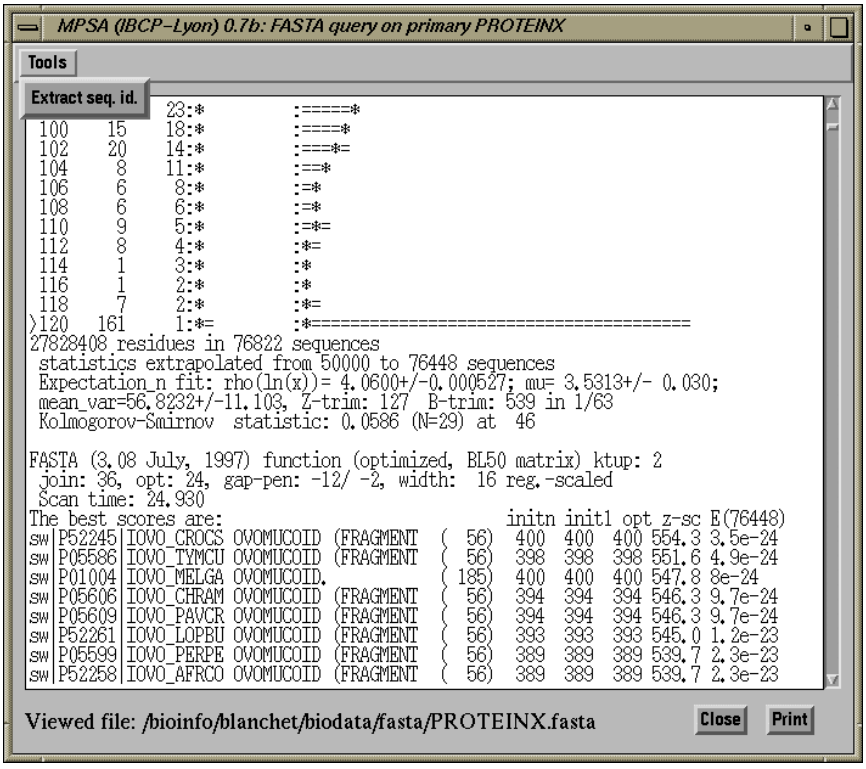


Figure 29 : MPSA, affichage des résultats d'une recherche avec FASTA

4.5.5.2.2 Recherche à l'aide d'un site ou d'une signature fonctionnelle : PattInProt

Parfois l'identification d'une famille de protéines à l'aide d'une séquence n'est pas possible car la similarité qui lie la famille n'apparaît pas au vu des séquences complètes mais à travers un motif protéique particulier. Ces motifs ont été répertoriés pour de nombreuses familles de protéines dans des banques comme PROSITE, Blocks, Pfam ou PRINTS (*cf.* 2.2.4.1).

Nous avons retenu la syntaxe de la banque PROSITE comme format des motifs à soumettre à PattInProt. Deux cas sont possibles : soit rechercher un motif connu et extrait de PROSITE, soit

construire un motif à l'aide d'un alignement multiple et de données biologiques indiquant certains acides aminés conservés ou essentiels à la famille de protéines.

4.5.5.2.2.1 Construction automatique d'un motif à partir d'un alignement

MPSA propose de construire un motif automatiquement à partir d'un alignement multiple. Le biologiste sélectionne tout ou partie de l'alignement à l'aide de la souris. Il choisit ensuite l'option « Primary➔PATTern IN PROTeins » (Figure 27) pour activer la méthode PattInProt. MPSA construit chaque position du motif en fonction de la colonne correspondante de l'alignement. Il reconnaît notamment les cas où la même position est répétée plusieurs fois de suite et ceux où l'importance biologique de la position relève plus de l'exclusion de certains acides aminés que de la présence des autres.

4.5.5.2.2.2 Critères de la recherche

La recherche peut être effectuée dans une banque de séquences comme nous l'avons décrit pour FASTA et BLAST. Mais elle peut aussi se dérouler dans l'alignement multiple ou les séquences affichées dans la fenêtre principale de MPSA.

Cette recherche se fera suivant un critère d'erreur par rapport au motif cherché déterminé par l'utilisateur. Ce critère peut prendre les valeurs suivantes :

- aucune erreur, identité stricte entre le motif cherché et le motif trouvé,
- un nombre discret[78] de positions fausses, le motif potentiel est rejeté s'il comporte n+1 positions différentes avec le motif cherché, n est fixé par l'utilisateur,
- un taux de similarité minimal, le motif potentiel est rejeté si son taux de similarité avec le motif cherché est en deça d'un seuil fixé par l'utilisateur. Cette option utilise un calcul des pénalités que nous avons mis au point. Il permet de privilégier la conservation des positions biologiquement importantes du motif lors d'une recherche avec erreur autorisée.

4.5.5.2.2.3 Analyse des résultats

Le fichier écrit par PattInProt après la recherche est affiché dans le visualisateur de fichier. Et le biologiste peut extraire les séquences contenues dans ce fichier en suivant la même procédure que pour FASTA et BLAST.

4.5.5.2.3 Matrice de points - dot plot

La matrice de points est une méthode graphique pour matérialiser la similarité de deux séquences. C'est une méthode exhaustive car elle affiche toutes les zones de similarités. Le graphique obtenu est une matrice avec des diagonales en fonction de la similarité locale des deux séquences (Figure 30). Un point P_{ij} d'une diagonale représente le score de similarité obtenu entre la position i de la séquence I et la position j de la séquence J. Une fenêtre de longueur 'n' est déplacée du début à la fin de la séquence I et la même fenêtre est aussi déplacée sur la séquence J. À chaque déplacement de l'une des deux fenêtres un score de similarité est calculé entre les deux segments. Si ce score est supérieur à un certain seuil, un point P_{ij} est affiché. Et les positions i et j correspondent à l'acide aminé

[78] Au sens mathématique du terme.

occupant le centre de ces segments respectivement sur la séquence I et sur la séquence J. Le score de similarité entre les deux segments est calculé à l'aide d'une matrice de similarité des acides aminés comme par exemple les matrices PAM, BLOSUM ou simplement la matrice identité.

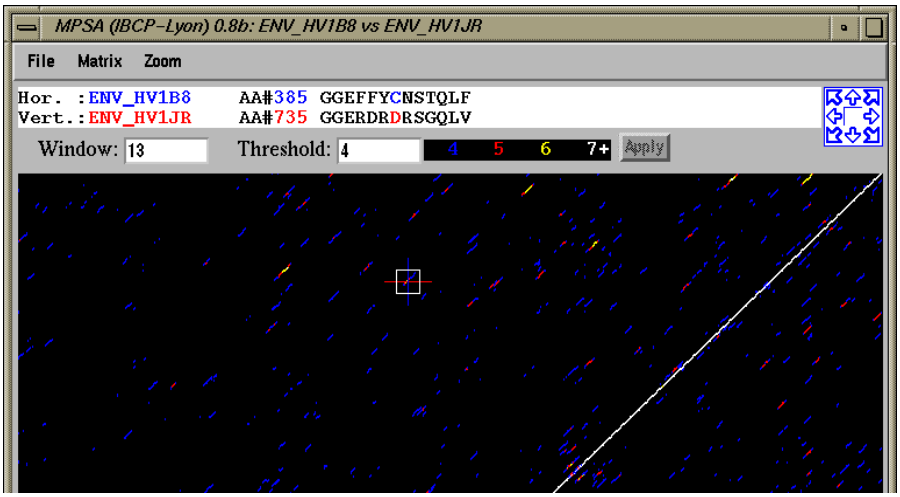


Figure 30 : MPSA, la matrice de points (« dotplot »)

La méthode du dot-plot implémentée dans MPSA offre au biologiste le choix de tous les paramètres : taille de la fenêtre, seuil d'affichage, matrice de similarité utilisée (Figure 30). Nous proposons également d'autres fonctionalités comme un zoom, un pavé de déplacement fin, une impression au format PostScript. Nous proposons également un niveau d'information supplémentaire en affichant les diagonales de similarité avec un code à 4 couleurs. Celles-ci représentent les valeurs s, s+1, s+2 et s+3 du score de similarité où s est le seuil d'affichage d'un point.

4.5.5.3 Alignement multiple

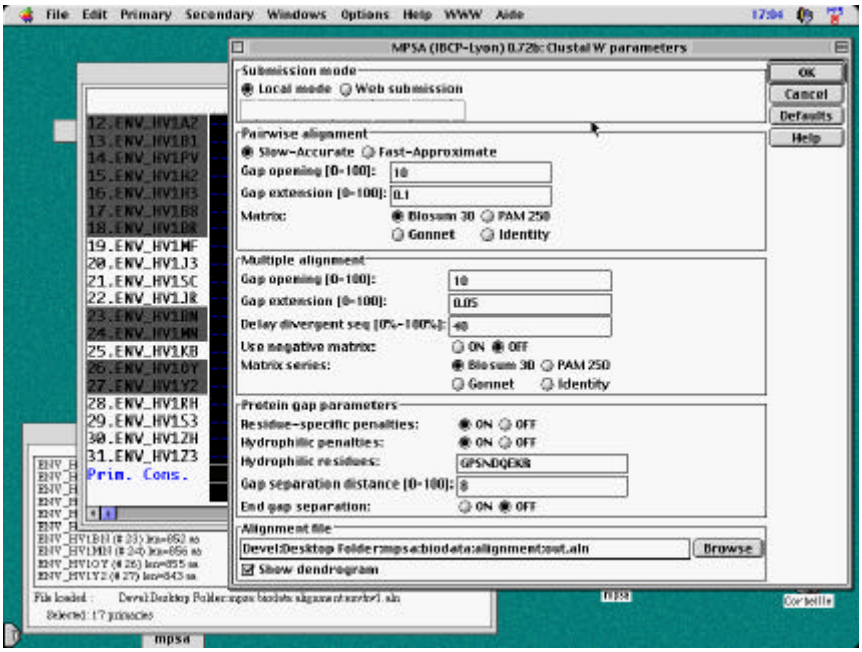


Figure 31 : MPSA, paramétrage d'un alignement multiple

Deux méthodes d'alignement multiple sont disponibles dans MPSA : Clustal W (P 180) et Multalin[79] (P 40). Elles sont accessibles au travers des menus « Primary➔Clustal W » et « Primary➔Multalin ». Lors de leur invocation une fenêtre de paramétrage (Figure 31) est proposée et parmi ces paramètres on retrouve par exemple les matrices de similarité, les valeurs de pénalisation des gaps ou le nom du fichier de sortie. Le calcul sera effectué sur la machine locale ou sur un serveur distant suivant le choix de l'utilisateur. Une fois le calcul terminé, l'alignement multiple est affiché dans la fenêtre principale.

4.5.5.3.1 Visualisation de la qualité de l'alignement

La qualité d'un alignement affiché dans MPSA est matérialisé par un codage de couleur particulier qui met en évidence les régions conservées des séquences. En effet les acides aminés conservés sont affichés en rouge et les zones de forte et faible similarité en vert. Les portions de l'alignement ne correspondant à aucun de ces critères sont affichées en bleu.

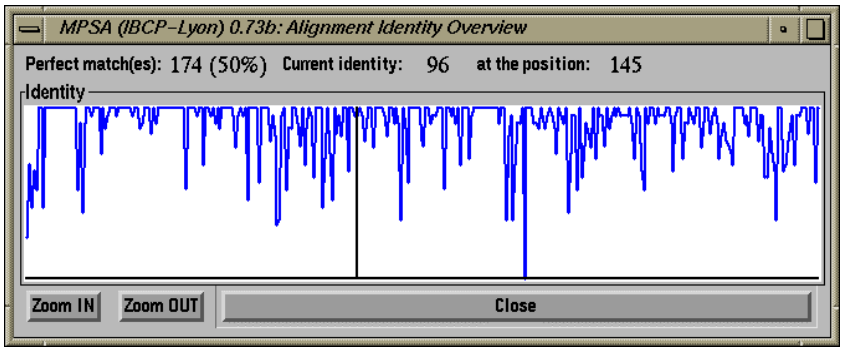


Figure 32 : MPSA, graphe du taux d'identité d'un alignement

Une autre manière d'appréhender la qualité d'un alignement est d'afficher la variation du taux d'identité des acides aminés le long de l'alignement. Deux façons de le faire sont proposées. La première consiste à tracer le graphe de ce taux d'identité en fonction de chaque position de l'alignement. Ce procédé est obtenu par le menu « Window➔Identity overview » qui provoque l'affichage de ce graphe dans une fenêtre graphique (Figure 32) avec toutes les fonctionnalités décrites au chapitre 4.5.3.6. Le second moyen consiste à afficher chaque colonne avec une couleur correspondant au taux d'identité des acides aminés de la colonne. Par exemple un dégradé avec onze couleurs du bleu au blanc correspondant aux intervalles [i ; i+0,09[avec i tel que $0 \leq i \leq 1$.

Tous ces codages de couleurs sont modifiables par l'utilisateur à travers le menu « Options➔Colors ».

4.5.5.3.2 Amélioration manuelle de l'alignement

La modification manuelle permet d'introduire dans un alignement des données biologiques obtenues par des méthodes expérimentales. Elle est réalisée par des insertions ou des suppressions dans l'alignement. L'outil « gap manager » permet ces modifications pour un nombre variable de gaps aux positions indiquées à l'aide de la souris (Figure 33). Des fonctions d'annulation, touches « undo » et « reset », permettent de revenir à un état antérieur si l'alignement obtenu est de moins bonne qualité que le précédent. Tous les codages de couleurs associés à la qualité de l'alignement sont évidemment

[79] Multalin, http://www.toulouse.inra.fr/multalin.html

actualisés à chaque insertion/suppression. De la même manière les structures associées à chaque acide aminé d'une séquence (états conformationnels, antigénicité, accessibilité au solvant,…) sont aussi décalées lors de ces modifications des séquences.

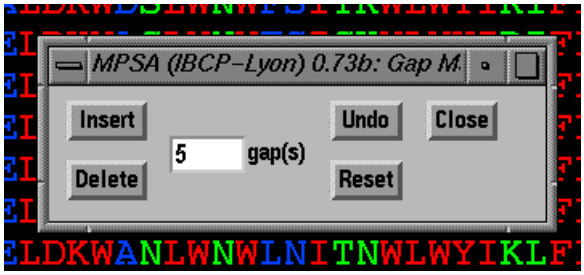


Figure 33 : MPSA, outil d'édition d'un alignement (« gap manager »)

4.5.5.3.3 Sauvegarde des alignements

Les alignements peuvent être sauvegardés soit pour être archivés soit pour être intégrés à une publication ou à une présentation. Dans le second cas la mise en forme et la qualité graphiques sont importantes, mais dans le premier cas seul importe l'exhaustivité des données enregistrées.

4.5.5.3.3.1 Fichiers d'archivages

La sauvegarde de l'alignement affiché s'effectue par le menu « File➔Save as » (Figure 26). Les formats de fichiers d'archivage disponibles dans MPSA sont le format Clustal W et le format MPSA. La sauvegarde peut également être réalisée sans conserver l'alignement des séquences en pratiquant une sauvegarde sous la forme d'un fichier de séquences.

4.5.5.3.3.2 Avec mise en forme pour une publication, présentation ou impression

La sauvegarde avec mise en forme d'un alignement permet de conserver les couleurs utilisées à l'affichage. Par contre il n'est plus possible de travailler dans MPSA avec un tel fichier. Ce type de fichier devra être exclusivement destiné à une impression ou à l'inclusion dans un logiciel de traitement de texte.

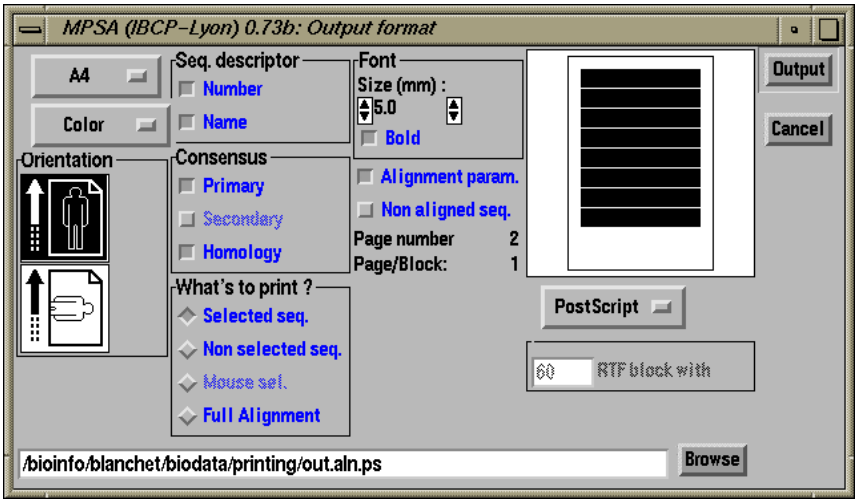


Figure 34 : MPSA, la fenêtre de sauvegarde avec mise en forme.

La sauvegarde commence avec l'utilisation du menu « File➔Print » qui active la fenêtre de paramétrage correspondante (Figure 34). Le biologiste choisit la mise en page (format du papier,

orientation portrait ou paysage,…) et les données à sauvegarder (identificateurs de séquence, séquence consensus, la sélection ou tout l'alignement,…). Il choisit également le format du fichier de sauvegarde (Annexe G) : PostScript ou RTF. Dans le cas d'une sauvegarde RTF, le biologiste choisit la largeur maximale des blocks de l'alignement. Dans le cas d'une impression PostScript, une petite fenêtre de pré-visualisation donne un aperçu de la disposition des blocks dans une page.

4.5.5.4 Recherche de sites ou signatures fonctionnelles : ProScan

La recherche de motifs protéiques est réalisée dans les séquences affichées dans la fenêtre principale et sélectionnées. Elle utilise l'algorithme que nous avons mis au point (*cf.* 1.1). Elle est initiée par la sélection du menu « Primary➔PROSITE scan ». Les paramètres disponibles dans la fenêtre qui apparaît (Figure 35) sont les suivants : (i)le choix entre un calcul local et une requête sur un serveur Web distant, (ii) la localisation des fichiers PROSITE sur le disque, (iii) le critère de sélection des motifs potentiels qui prend les mêmes valeurs que ceux de la méthode PattInProt (*cf.* 4.5.5.2.2.2). Les résultats obtenus à l'issue du calcul sont affichés dans le visualisateur de fichier.

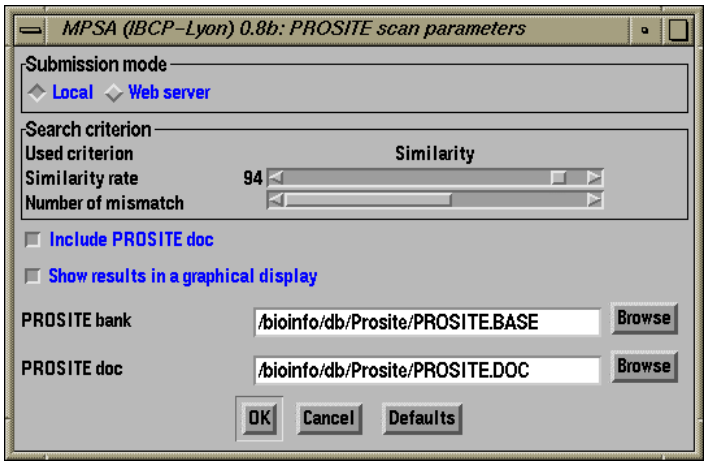


Figure 35 : MPSA, la recherche dans une séquence des occurrences des motifs répertoriés dans PROSITE

4.5.5.5 Incorporation de structures secondaires

MPSA permet d'incorporer des structures secondaires dans un alignement. Ces structures secondaires sont observées (déduites d'un fichier PDB par DSSP (P 112) ou P-SEA (P 121)) ou prédites à l'aide de différentes méthodes.

4.5.5.5.1 Les méthodes de prédiction de structure secondaire disponibles

En mode local, les méthodes de prédiction de structure secondaire disponibles dans MPSA sont celles implémentées dans la bibliothèque « biolcp » : GOR 1 (P 74), GOR 2 (P 83) et Levin (P 126). En mode connecté à un serveur Web distant, les méthodes sont celles disponibles sur ce serveur, par exemple celles du serveur NPS@[80] (*cf.* 4.2.5). Le choix d'un de ces deux modes, ainsi que le choix de la méthode sont faits dans la fenêtre de prédiction de structure secondaire Figure 36. Les méthodes locales proposent leurs paramètres de calcul, mais l'invocation des méthodes « distantes » se fait en utilisant les paramètres par défaut propres à chacune d'elles.

[80] http://pbil.ibcp.fr/NPSA

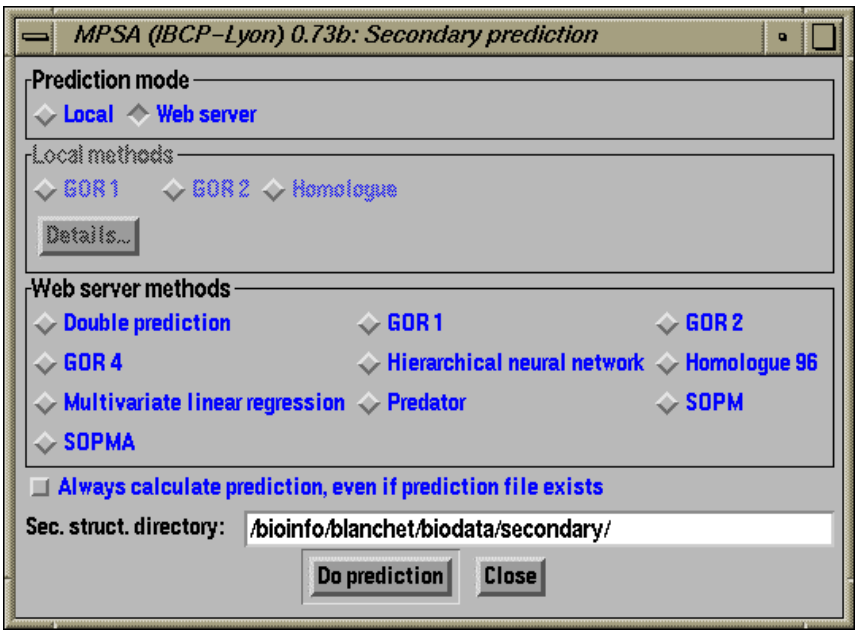


Figure 36 : MPSA, fenêtre des paramêtres des prédictions de structure secondaire

Quel que soit le mode de calcul (local ou distant), la méthode choisie est appliquée sur les séquences sélectionnées dans la fenêtre principale, qu'elles soient alignées ou non. Si plusieurs séquences sont prédites, un moniteur graphique indique au biologiste l'avancement des calculs.

4.5.5.5.2 Visualisation des structures secondaires dans l'affichage des séquences

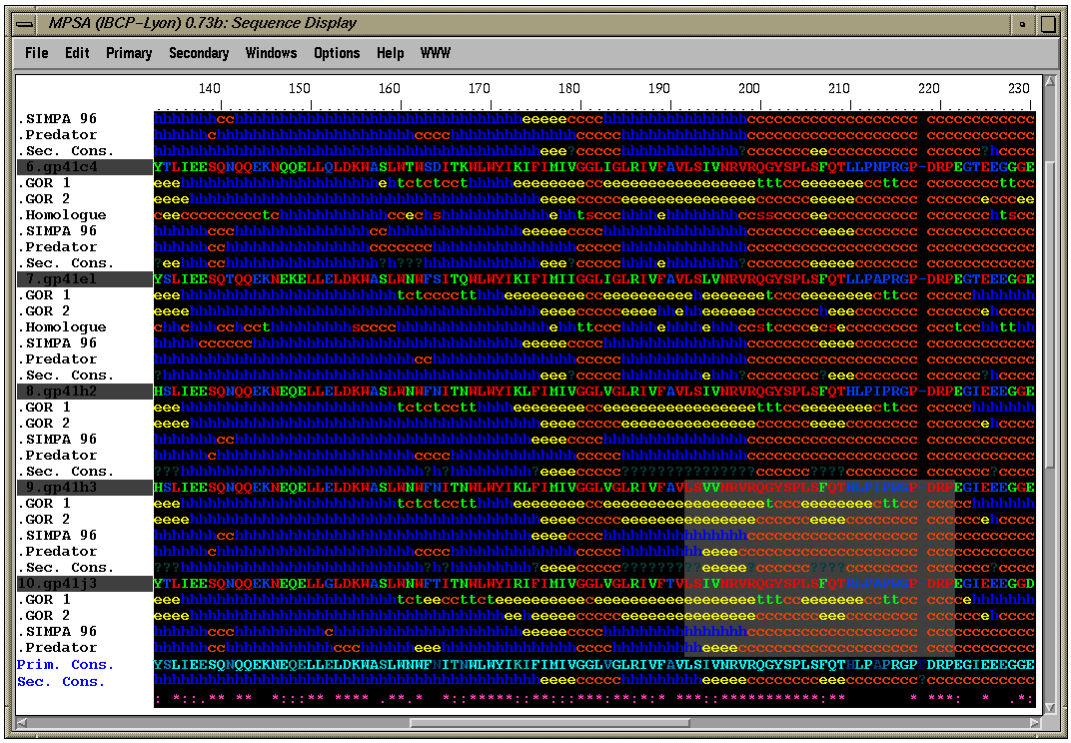


Figure 37 : MPSA, incorporation de structures secondaires dans l'affichage des séquences.

Une fois les calculs terminés les fichiers contenant les prédictions de structure secondaire sont stockés sur le disque local[81]. MPSA ouvre alors automatiquement ces fichiers et en extrait les données. Les données peuvent également être extraites de fichiers existants : prédictions calculées précédemment ou structures secondaires issues de données structurales. Les formats reconnus sont les formats DSSP, PHD, PREDATOR et MPSA.

Les structures secondaires extraites des fichiers sont alors incluses à l'affichage des séquences dans la fenêtre principale sous la forme d'un résumé alphabétique coloré (Tableau 27). Ce résumé se présentent sous la forme d'une séquence de 'h', 'e', 't' et 'c'. Ces lettres peuvent être en bas de casse (minuscule) ou en haut de casse (majuscule). Cette différence traduit la qualité de la prédiction de l'état conformationnel pour cet acide aminé : minuscule ➔ faible qualité ou indéterminée ; majuscule ➔ grande qualité. En effet, certaines méthodes comme PHD incluent cette estimation dans le fichier résultat.

Tableau 27 : MPSA, code alphabétique coloré des structures secondaires.

État conformationnel	**Code[82]**	**Couleur**
Hélice alpha	H ou h	Bleu
Feuillet béta	E ou e	Jaune
Coude	T ou t	Vert
Structure apériodique	C ou c	Orange

4.5.5.5.3 Visualisation du détail de la prédiction

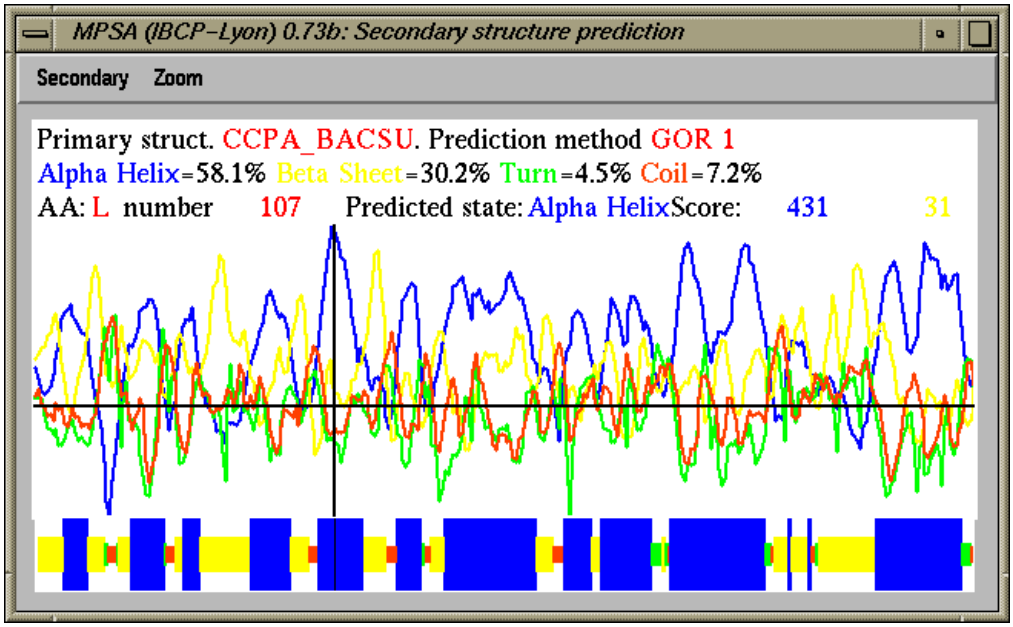


Figure 38 : MPSA, vue de détail d'une prédiction de structure secondaire.

Afin de pouvoir estimer la qualité d'une prédiction, le biologiste affiche la vue de détail (Figure 38) en double-cliquant sur l'identificateur de cette prédiction. Cette vue de détail est constituée

[81] Ce qui a pour avantage de pouvoir utiliser ces prédictions lors de sessions ultérieures.
[82] En minuscule ou en majuscule suivant la qualité de la prédiction.

des graphes des scores pour chaque état conformationnel ainsi que d'un résumé sous la forme de boites de tailles et de couleurs différentes suivant l'état conformationnel prédit.

Les paramètres locaux retournés par le curseur sont le code IUPAC à une lettre de l'acide aminé, sa position dans la séquence, l'état conformationnel prédit et les scores pour chaque état conformationnel proposé par la méthode. Ils s'ajoutent aux paramètres généraux que sont l'identificateur de la séquence prédite, la méthode de prédiction et les paramètres de calcul utilisés (lorsqu'ils sont disponibles), et la fréquence de chaque état conformationnel dans toute la prédiction.

4.5.5.6 Prédictions de caractéristiques physico-chimiques

MPSA propose également des prédictions de caractéristiques physico-chimiques. Avec le menu « Primary➔Physico-chemical profiles ». Les méthodes proposées (Figure 39) sont celles implémentées dans la bibliothèques « biolcp » (Tableau 19).

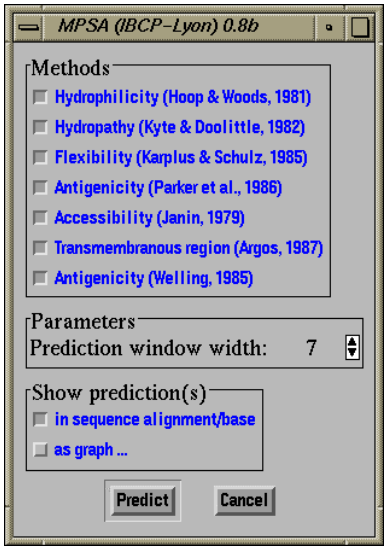


Figure 39 : MPSA, les prédictions de caractéristiques physico-chimiques

Le biologiste applique ces méthodes aux séquences sélectionnées dans la fenêtre principale. Il choisit quelles méthodes appliquer et également le mode d'affichage des résultats. Ceux-ci peuvent être visualisés sous la forme de résumés symboliques et sous la forme de graphes. Le résumé est affiché dans la fenêtre principale sous la séquence prédite. Le codage utilisé est le suivant : un '+' s'affiche si la valeur à cette position est supérieure à un seuil fixé par l'utilisateur au moyen du menu « Options➔Set physico-chemical thresholds ». Les graphes sont affichés dans des fenêtres de graphes de MPSA et disposent de toutes les fonctionnalités décrites auparavant (*cf.* 4.5.3.6).

4.5.5.7 Édition et manipulation des séquences

Les séquences des protéines sur lesquelles travaille l'utilisateur sont affichées dans la fenêtre principale (Chap. 4.5.3.2.2). Le biologiste peut très simplement modifier une de ces séquences en « mutant » un ou plusieurs acides aminés ou en insérant/supprimant des segments. Ces modifications s'effectuent dans l'éditeur de séquences (Figure 40). Cet outil est disponible dans le menu « Primary➔Edit current ». Il permet également de dupliquer la séquence en modifiant son nom. Lorsque les séquences sont alignées, les modifications sont réalisées par insertion ou suppression de gaps comme nous l'avons décrit précédemment avec l'outil « gap manager » (Figure 33).

Figure 40 : MPSA, l'éditeur de séquences d'acides aminés

Afin de rassembler côte à côte des séquences appartenant à la même famille, le biologiste peut déplacer des séquences à l'aide de la souris. Il doit tout d'abord sélectionner les séquences qu'il désire déplacer. Il opère cette sélection en cliquant sur le nom de ces séquences avec la souris. Ensuite il clique sur ces séquences sélectionnées et déplace le pointeur de la souris vers l'endroit où il veut déposer ces séquences, tout en maintenant le bouton de la souris enfoncé. Évidemment si les séquences déplacées avaient des structures associées (structures secondaires, caractéristiques physico-chimiques), celles-ci sont déplacées aussi. De la même manière le biologiste peut également déplacer des structures secondaires ou des caractéristiques physico-chimiques associées à une même séquence.

4.5.5.8 Autres outils disponibles

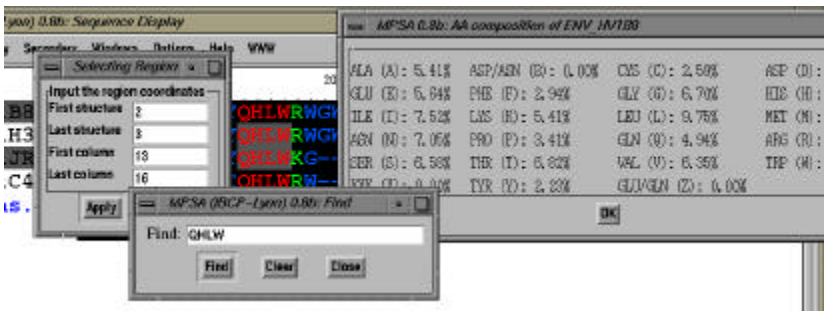


Figure 41 : MPSA, différents outils : la recherche d'une chaine, la sélection d'une région d'un alignement et la composition d'une séquence.

De nombreux outils sont disponibles dans MPSA afin d'augmenter l'efficacité et de simplifier l'analyse de séquences de protéine. Parmi lesquelles nous pouvons citer les suivants :

Menu	**Actions**
File➔File viewer	Affiche n'importe quel fichier texte
Edit➔Find	Recherche stricte d'une chaine de caractères dans les séquences
Edit➔Select a region	Sélection d'une région en entrant ses coordonnées en numéro de séquence et position dans la séquence
Primary➔AA composition	Donne la composition en acides aminés des séquences sélectionnées

Menu	**Actions**
Options➔Style	Modification de la taille et du style de la police utilisée pour les séquences
Help➔About MPSA	Version utilisée de MPSA, site Web et adresse mél[83] de contact

4.5.6 Mise à disposition de la communauté scientifique : le site Web de MPSA

Afin de proposer le logiciel MPSA le plus simplement à la communauté scientifique, nous avons mis en place un site Web dédié à MPSA. L'URL est le suivant : http://www.ibcp.fr/mpsa. La page d'accueil (Figure 42) oriente le visiteur sur d'autres pages suivant l'information qu'il désire obtenir.

Figure 42 : le site Web de MPSA : la page d'accueil

Ainsi le biologiste peut télécharger MPSA dans le format d'exécutable utilisable sur sa machine qui peut être une station de travail sous UNIX ou un ordinateur individuel. La page de téléchargement est accessible à l'adresse http://www.ibcp.fr/mpsa/download.html. Différents liens hypertextes permettent de choisir l'archive correspondant au système d'exploitation désiré (Figure 43). Depuis octobre 1998, 416 récupérations de l'archive MPSA ont été effectuées (Mars 1999).

Une documentation est également disponible au format HTML à l'adresse http://www.ibcp.fr/mpsa/doc ou par téléchargement au format Word® ou PostScript. Le biologiste peut ainsi consulter les différentes rubriques de l'aide de MPSA. Chaque rubrique est affichée dans une page Web différente (Figure 43), et un sommaire permet de choisir la rubrique désirée.

[83] Mél : Messagerie ÉLectronique

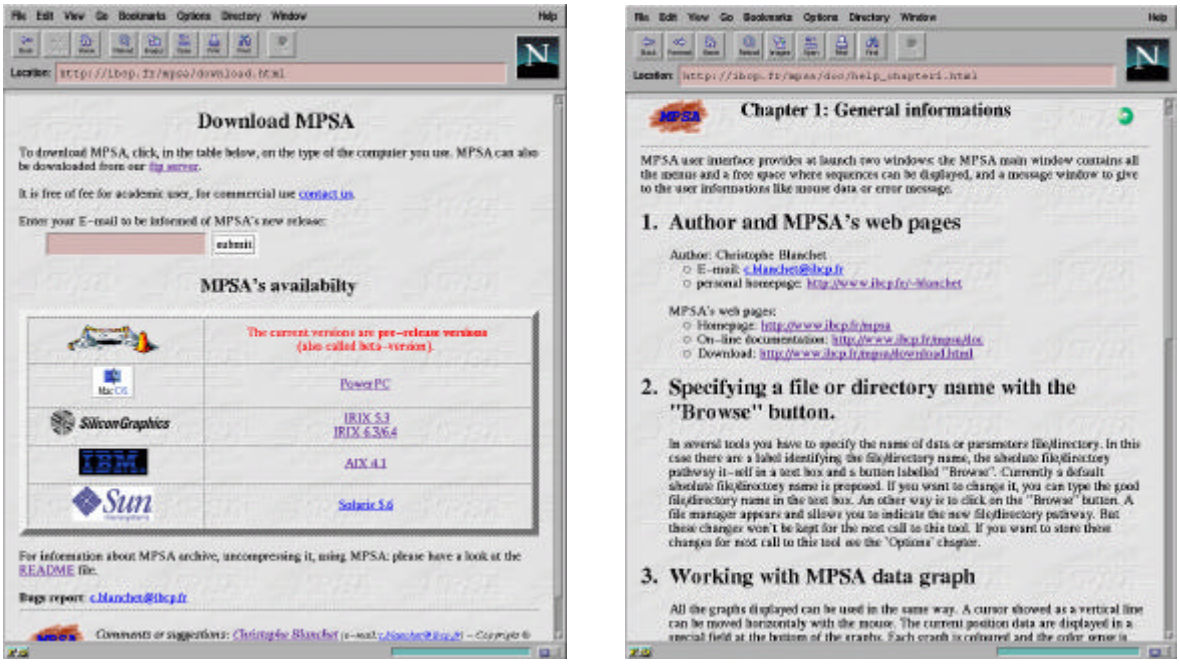


Figure 43 : le site Web de MPSA, la page de téléchargement et le chapitre 1 de la documentation.

Le biologiste peut également s'inscrire sur la « user mailing list » de MPSA : il lui suffit d'indiquer son adresse mél[84] dans le champ correspondant. Et par la suite il sera informé de la mise à disposition des nouvelles versions de MPSA, des avis de bogues, *etc.* par un courrier électronique. Ce qui lui évite de faire une veille systématique des nouvelles fonctionnalités de MPSA. Depuis janvier 1999, 83 utilisateurs de MPSA se sont inscrit sur la « user mailing list » de MPSA (fin mars 1999).

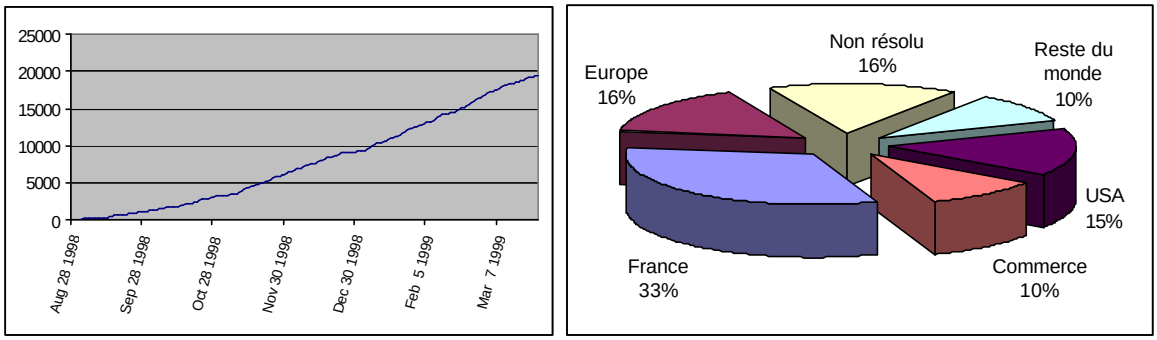


Figure 44 : le site Web MPSA, évolution et répartition des connexions

[84] Aucune autre information n'est demandée.

5 Conclusion

Nos différents développements proposent aux biologistes un environnement logiciel d'analyse de séquences de protéines. Cet environnement dispose notamment de fonctionnalités client-serveur qui déchargent le biologiste de la gestion des bases de données qu'il utilise : téléchargement, mise à jour, concordance des formats de fichiers, *etc*. Les analyses de séquences peuvent être effectuées sur des serveurs distants à l'aide de clients Web génériques (Netscape, Internet Explorer®,…) en interrogeant notre serveur Web NPS@. Le biologiste dispose également des fonctionnalités client-serveur de MPSA. Ce logiciel lui procure de nombreux outils biologiques et interactifs pour conduire son analyse.

Ces deux moyens d'analyse, NPS@ et MPSA, répondent à des besoins différents des biologistes. NPS@ procure une analyse efficace avec de nombreuses méthodes et une connexion très forte avec les banques de données mondiales grâce aux liens hypertextes. De plus d'autres sites Web d'analyse de protéines référencent NPS@ en extension des méthodes qu'ils proposent. MPSA permet une analyse tout aussi efficace puisque les méthodes de NPS@ lui sont accessibles. Il procure en plus au biologiste une interactivité et une convivialité que seul un logiciel intégré et spécialisé peut lui apporter. Cependant, MPSA nécessite pour l'instant un serveur Web compatible avec son mode d'interrogation.

Afin que cette compatibilité d'un serveur Web avec MPSA puisse être réalisée rapidement et simplement, nous proposons des interfaces de serveur entre MPSA et plusieurs méthodes standards d'analyse de séquences de protéine : BLAST, FASTA, Clustal W,… Ces interfaces sont des scripts en langage Perl à installer dans le répertoire « cgi-bin » du serveur Web. De plus la bibliothèque *biolcp* met à la disposition des bioinformaticiens les routines « client » d'invocation d'un serveur Web compatible MPSA. Ils peuvent alors les utiliser dans les logiciels qu'ils développent. Ces logiciels seront alors capables de soumettre des requêtes d'analyse à un serveur Web compatible MPSA. Ainsi la disponibilité à la fois du client et du serveur MPSAweb permet l'installation rapide d'outils intranet d'analyse de séquences de protéine.

Internet est victime de son succès. Il mérite alors doublement son surnom « d'autoroutes de l'information » puisqu'il connaît lui aussi des périodes de « bouchon » au cours desquelles son débit chute considérablement. C'est pourquoi nous avons intégré à MPSA plusieurs méthodes d'analyse en mode local. Parmi celles-ci se trouve la méthode de recherche de motifs protéiques que nous avons mise au point. Cet outil de recherche essaie d'apporter une solution satisfaisante au paradoxe suivant : autoriser des erreurs lors d'une recherche de motifs protéiques tout en conservant dans les motifs trouvés la majeure partie de l'information biologique. Cette autorisation d'erreurs est nécessaire pour identifier tous les motifs potentiels qui ne sont pas encore répertoriés dans les banques de données au moment de la recherche.

Ce décalage entre la publication des séquences et leur annotation complète dans les banques de données ne va que s'accentuer dans les prochaines années. En effet les programmes de séquencage complet de génomes se multiplient et sont menés à terme de plus en plus rapidement. Les données brutes disponibles augmentent exponentiellement et imposent une annotation rapide, la plus exhaustive et avec le moins d'erreurs possible. Car il est unanimement reconnu que les annotations des séquences dans les banques contiennent des erreurs. Et il est aussi unanimement reconnu que ce taux d'erreur serait très difficile voire impossible à estimer (P 114). Un moyen pour diminuer le risque d'erreur serait de simplifier l'annotation. Dans cette optique, une perspective pour MPSA serait de lui

ajouter un module d'annotation de séquences de protéine dans les formats des banques mondiales comme SWISS-PROT et TrEMBL.

Une autre perspective pour MPSA serait de lui adjoindre des modules dédiés aux acides nucléiques et des modules 3D. Les premiers permettraient de travailler directement sur les données issues des programmes de séquençage de génome. Ces modules acides nucléiques seront dans un premier temps des outils de traduction auquels nous pourrons adjoindre ensuite des modules existants comme par exemple des modules de recherche de phase ouverte de lecture (ORF). Les modules 3D proposeraient dans un premier temps une interface sous forme de scripts RasMol (URL 44) puis à plus long terme des fonctionnalités plus complexes comme par exemple de la phylogénie structurale. MPSA serait ainsi capable d'effectuer, lorsque les données disponibles sont suffisantes, une analyse du gène à la structure tridimensionnelle de la protéine codée par ce gène, voire à sa fonction.

Nous avons évoqué précédemment les fonctionnalités client-serveur de MPSA. Une évolution de celles-ci serait d'incorporer à MPSA la technologie des objets (P 28). En effet des interfaces objets se mettent en place au niveau de serveurs Web afin de proposer une interface standard objet à différents outils biologiques (P 19, URL 10). MPSA pourrait alors grâce à l'incorporation de ces objets étendre efficacement ses champs d'application. Notamment l'ajout à MPSA d'un module client utilisant le standard CORBA[85] de l'OMG (URL 32) est tout à fait envisageable puisque l'IDL[86] est proche du langage C++ et peut également être invoqué en langage C. Dans le même but, la mise en place au sein de NPS@ d'un serveur CORBA, ou ORB[87], dédié à l'analyse de séquences de protéine serait l'occasion de définir et de proposer des interfaces client-serveur standards à des méthodes comme la recherche de motifs protéiques ou la prédiction de structure secondaire. Ces perspectives s'accorderont avec les projets bioinformatiques actuellement en cours que sont BioPerl (URL 6), l'interface serveur CORBA de l'EBI (http://corba.ebi.ac.uk) et la définition du langage BIOML (URL 5). Tous ces développements futurs s'intégreront dans un contexte plus général de Pôle Bioinformatique Lyonnais en place depuis janvier 1998.

[85] CORBA : Common Object Request Broker Architecture
[86] IDL : Interface Definition Langage
[87] ORB : Object Request Broker

6 Références

6.1 Références bibliographiques

P 1 Abola EE, Manning NO, Prilusky J, Stampf DR, Sussman JL (1996), The Protein Data Bank : current status and future challenges. *J. Res Natl Inst Stand technol* **101**, 231-241

P 2 Achard F, Cussat-Blanc C, Viara E, Barillot E (1998) The new Virgil database : a service of rich link. *Bioinformatics*. **14**, 342-348.

P 3 Achard F, Dessen P (1998) GenXref VI : automatic generation of links between two heterogeneous databases. *Bioinformatics*. **14**, 20-24.

P 4 Achard F, Vaysseix G, Dessen P, Barillot E (1999) Virgil database for rich links (1999 update). *Nucleic Acids Res*. **27**, 113-114.

P 5 Agarwal P, States DJ (1998) Comparative accuracy of methods for protein sequence similarity search. *Bioinformatics* **14**, 40-7.

P 6 Altschul SF (1998) Fundamentals of database searching. *Trends Guide to Bioinformatics*. 7-9.

P 7 Altschul SF, Gish W (1996) Local alignment statistics. *Methods Enzymol.* **266**, 460-480.

P 8 Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. *J. Mol. Biol*. **215**, 403-410

P 9 Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman D (1997) Gapped BLAST and PSI-BLAST : a new generation of protein database search programs. *Nucelic Acids Res.* **25**, 3389-3402

P 10 Anfinsen CB (1973) Principles that govern the folding of protein chains. *Science* **181**, 223-230

P 11 Appel RD, Bairoch A, Hochstrasser DF (1994) A new generation of information retrieval tools for biologists: the example of the ExPASy WWW server. *Trends Biochem Sci*. **19**, 258-260.

P 12 Attwood TK, Findlay JBC (1994) PRINTS - A protein motif fingerprint database. *Protein Eng.* **7**, 195-203.

P 13 Attwood TK, Flower DR, Lewis AP, Mabey JE, Morgan SR, Scordis P, Selley JN, Wright W (1999) PRINTS prepares for the new millennium. *Nucleic Acids Res.* **27**, 220-225.

P 14 Bairoch A (1991) SEQANALREF : a sequence analysis bibliographic reference databank. *Comput Applic Biosci*. **7**, 268.

P 15 Bairoch A (1999) The ENZYME data bank in 1999. *Nucleic Acids Res.* **27**, 310-311.

P 16 Bairoch A, Apweiler R (1999) The SWISS–PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Res*. **27**, 49-54

P 17 Bairoch A, Bucher P (1994) PROSITE: recent developments. *Nucleic Acids Res.* **22**, 3583-3589.

P 18 Baker PG, Brass A (1998) Recent developments in biological sequence database. *Curr Opin Biotechnol*. **9**, 54-58.

P 19 Barillot E, Leser U, Lijnzaad P, Cussat-Blanc C, Jungfer K, Guyon F, Vaysseix G, Helgesen C, Rodriguez-Tomé P (1999) A proposal for a standard CORBA interface for genome maps. *Bioinformaticsl.* **15**, 157-169

P 20 Barker WC, Garavelli JS, McGarvey PB, Marzec CR, Orcutt BC, Srinivasarao GY, Yeh LSL, Ledley RS, Mewes HW, Pfeiffer F, Tsugita A, Wu C (1999) The PIR-international protein sequence database. *Nucleic Acids Res*. **27**, 39-43.

P 21 Barker WC, Pfeiffer F, George DG (1996) Superfamily Classification in PIR-International Protein Sequence Database. *Methods Enzymol*. **366**, 59-71.

P 22 Bateman A, Birney E, Durbin R, Eddy SR, Finn RD, Sonnhammer ELL (1999) Pfam 3.1: 1 313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Res.* **27**, 260-262.

P 23 Benson DA, Boguski MS, Lipman DJ, Ostell J, Ouellette BFF, Rapp BA, Wheeler DL (1999) Genbank. *Nucleic Acids Res*. **27**, 18-24.

P 24 Berners-Lee TJ, Cailliau R, Groff JF, Pollermann B (1992) World-Wide Web : the information universe. *Electronic Networking : Research, Application and Policy*. **2**, 52-58.

P 25 Bernstein FC, Koetzle TF, Williams GJB, Meyer Ef, Brice MD, Rodgers JR, Kennard O, Shimanouchi T, Tasumi M (1977) The Protein data Bank : a computer based archival file for macromolecular structures. *J Mol Biol*. **112**, 535-542.

P 26 Biou V, Gibrat JF, Levin JM, Robson B, Garnier J (1988) Secondary structure prediction : combination of three different methods. *Prot. Eng.* **2**, 185-191

P 27 Blundell TL, Jonhson LN (1976) Protein Crystallography. *Academic Press, New–York.*

P 28 Bouzeghoub M, Gardarin G, Valduriez P (1998) Les OBJETS, Edition revue et augmentée. Eyrolles, Paris. 450p.

P 29 Boyle J (1998) A visual environment for the manipulation and integration of JAVA Beans. *Bioinformatics*. **14**, 739-748.

P 30 Brenner SE (1998) Practical database searching. *Trends Guide to Bioinformatics*. 9-12.

P 31 Briffeuil P, Baudoux G, Lambert C, De Bolle X, Vinalls C, Feytmans E, Depeyrieux E.(1998) Comparative analysis of seven multiple protein sequence alignment servers : clues to enhance reliabillity of predictions. *Bioinformatics*, 14, 357-366.

P 32 Brownstein MJ, Trent JM, Boguski MS (1998) Functional genomics. *Trends Guide to Bioinformatics*. 27-29.

P 33 Bucher P, Hoffman K (1996) A sequence similarity algorithm based on a probalistic interpretation of an alignment scoring system.*Ismb*. **4**, 44-51.

P 34 Carrillo H, Lipman DJ (1988) The multiple sequence alignment problem in biology. *SIAM J Appl Math*. **48**, 563-598.

P 35 Chaléat P, Charnay D (1996) HTML et la programmation de serveurs. Eyrolles, Paris. 264p.

P 36 Chen CC, Ingh JP, Altman RB (1999) Using imperfect secondary structure predictions to improve molecular structure computations. *Bioinformatics*. **15**, 53-65.

P 37 Chervitz SA, Hester ET, Ball CA, Dolinski K, Dwight SS, Harris MA, Juvik G, Malekian A, Roberts S, Roe T, Scafe C, Schroeder M, Sherlock G, Zhu SWY, Cherry JM, Botstein D (1999) Using the saccharomyces genome database (SGD) for analysis of protein similarities and structure. *Nucleic Acids Res*. **27**, 74-78.

P 38 Chou PY, Fasman GD (1974), Conformational parameters for amino acids in helical, b-sheet and random coil regions calculated from proteins. *Biochemistry* **13**, 211-222

P 39 Chou PY, Fasman GD (1974), Prediction of protein conformation. *Biochemistry* **13**, 222-245

P 40 Corpet F (1988) Multiple sequence alignment with hierarchical clustering. *Nucl. Acids Res.* **16**, 10881-10890

P 41 Corpet F, Grouzy J, Kahn D (1998) The ProDom database of protein families. *Nucl. Acids Res.* **26**, 323-326

P 42 Corpet F, Grouzy J, Kahn D (1999) Recent improvments of the ProDom database of protein domain families. *Nucl. Acids Res.* **27**, 263-267

P 43 Cuff JA, Clamp ME, Siddiqui AS, Finlay M, Barton GJ (1998) JPred : a consensus secondary structure prediction server. *Bioinformatics* **14**, 892-3.

P 44 Danchin Antoine. La Barque de Dèlphes – Ce que révèle le texte des génomes. Paris, Odile Jacob (Sciences), 1998. 396p.

P 45 Davis L (1991). *The Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New–York.

P 46 Dayhoff MO (1976) The origin and evolution of protein superfamilies. *Fed Proc.* **35**, 2132-2138.

P 47 Dayhoff MO, Eck RV, Chang MA, Sochard MR (1965) *Atlas of protein sequence and structure* Vol. 1. National Biomedical Research Foundation, Silver Spring, MD.

P 48 Dayhoff MO, Eck RV, Park CM (1972) A model of evolutionary change in proteins. In Dayhoff MO (ed.), *Atlas of protein sequence and structure*. National Biomedical Research Foundation, Washington, DC, Vol. 5, p. 89-99.

P 49 Dayhoff MO, McLaughlin PJ, Barker WC, Hunt LT (1975). *Naturwissenschaften.* **62**, 154-161.

P 50 Dayhoff MO, Schwartz RM, Orcutt BC (1978) A model of evolutionary change in proteins. In Dayhoff MO (ed.), *Atlas of protein sequence and structure*. National Biomedical Research Foundation, Washington, DC, Vol. 5, Suppl. 2, p. 345-352.

P 51 Deleage G, Blanchet C, Geourjon C (1997) Protein structure prediction. Implication for the biologist. *Biochimie.* **79**, 681-686.

P 52 Deléage G, Clerc FF, Roux B, Gautheron DC (1988) ANTHEPROT : a package for protein sequence analysis using a microcomputer. *Comput Applic Biosci*. **4**, 351-356.

P 53 Deléage G, Roux B (1987) An algorithm for protein secondary structure prediction based on class prediction. *Prot. Eng.* **1**, 289-294

P 54 Deléage G, Roux B (1989) Use of class prediction to improve protein secondary structure prediction : Joint prediction with methods based on sequence homology. *In : Prediction of protein structure and the principles of protein conformation* (Fasman G, ed.) Plenum, New York and London, page 587

P 55 Depiereux E, Baudoux G, Briffeuil P, De Bolle X, Vinals C, Feytmans E (1997) Match-Box_server : a multiple sequence alignment tool placing emphasis on reliability. *Comput Applic Biosci*. **13**, 249-256.

P 56 Depiereux E, Feytmans E (1992) MATCH-BOX : a fundamentally new algorithm for the simultaneous alignment of several protein sequences. *Comput Applic Biosci*. **8**, 501-509.

P 57 Di Francesco V, Garnier J, Munson PJ (1996) Improving protein secondary structure prediction with aligned homologous sequences. *Protein Sci*. **5**, 106-113.

P 58 Di Francesco V, Munson PJ, Garnier J (1995) Use of multiple alignments in protein secondary structure prediction. *Processing of the 28th Annual Hawaï International Conference on System Sciences* **5**, 285-291

P 59 Discala C, Ninnin M, Achard F, Barillot E, Vaysseix G (1999) Dbcat : a catalog of biological databases. *Nucleic Acids Res.* **27**, 10-1

P 60 Donnelly D, Overington JP, Blundell TL (1994) The prediction and orientation of a helices from sequence alignments: then comined use of environment-dependent substituion tables, Fourier transform methods and helix capping rules. *Prot. Eng.* **7**, 645-653

P 61 Eisenhaber F, Persson B, Argos P (1995) Protein structure prediction : recognition of primary, secondary, and tertiary structural features from amino acid sequence. *Crit. Rev. Biochem. Mol. Biol.* **30**, 1-94

P 62 Ellis RJ, Hartl FU (1999) Principles of protein folding in the cellular environment. *Curr Opin Struct Biol*. **9**, 102-110

P 63 Etzold T, Argos P (1993) Transforming a set of biological flat file libraries to a fast access network. *Comput. Appl. Biosci*. **9**, 59-64.

P 64 Etzold T, Argos P (1993), SRS : an indexing and retrieval tool for flat file data libraries. *Comput Appl. Biosci*. **9**, 49-57

P 65 Fasman GD (1989) Development of the prediction of protein structure. *In: Prediction of protein structure and the principles of protein conformation* (Fasman G ed) Plenum, New York and London 193p

P 66 Fleischmann RD, Adams MD, White O, Clayton RA, Kirkness EF, Kerlavage AR, Bult CJ, Tomb JF, Dougherty BA, Merrick JM, *et al*., Venter JC (1995) Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science* **269**, 496-512.

P 67 Fraser CM, Gocayne JD, White O, Adams MD, Clayton RA, Fleischmann RD, Bult CJ, Kerlavage AR, Sutton G, Kelley JM, *et al*., Venter JC (1995) The minimal gene complement of Mycoplasma genitalium. *Science*. **269**, 397-403.

P 68 Friemann A, Schmitz S (1992) A new approach for displaying identities and differences among aligned amino acid sequences. *CABIOS*. **8**, 261-265.

P 69 Frishman D, Argos P (1996) Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence. *Protein Eng.* **9**, 133-142

P 70 Frishman D, Heumann K, Lesk A, Mewes HW (1998) Comprehensive, comprehensible, distributed and intelligent databases : current status. *Bioinformatics*. **14**, 551-561.

P 71 Garavelli JS (1999) The RESID database of protein structure modifications. *Nucleic Acids Res.* **27**, 198-199.

P 72 Garnier J (1990) Protein structure prediction. *Biochimie* **72**, 513-524

P 73 Garnier J, Gibrat J-F, Robson B (1996) GOR method for predicting protein secondary structure from amino acid sequence. *Methods Enzymol.* **266**, 540-553

P 74 Garnier J, Osguthorpe, DJ, Robson B (1978) Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *J. Mol. Biol.***120**, 97-120

P 75 Garratt RC, Taylor WR, Thornton, JM (1985) The influence of tertiary structure on secondary structure prediction. Accessibility versus predictability for beta structure. *FEBS Lett.* **188**, 59-62

P 76 George DG, Barker WC, Hunt LT (1986) The protein identification resource (PIR). *Nucleic Acids Res*. **14**, 11-15.

P 77 George DG, Hunt TL, Barker WC (1996) PIR-International Protein Sequence Database. *Methods Enzymol*. **366**, 41-59.

P 78 Geourjon C, Deléage G (1993) Interactive and graphic coupling between multiple alignments, secondary structure predictions and motif/pattern scanning into proteins. *Comput Applic Biosci*. **9**, 87-91.

P 79 Geourjon C, Deléage G (1994) SOPM: a self -optimised prediction method for protein secondary structure prediction. *Prot. Eng.* **7**, 154-164

P 80 Geourjon C, Deléage G (1995) ANTHEPROT 2.0 : A 3D module coupled to protein sequence analysis methods. *J. Mol. Graph.* **13**, 209-212

P 81 Geourjon C, Deléage G (1995) SOPMA: significant improvements in protein secondary structure prediction by consensus prediction from multiple alignments. *Comput. Appl. Biosci.* **11**, 681-684

P 82 Geourjon C, Deléage G, Roux B (1991) ANTHEPROT : an interactive graphics software for analyzing protein structures from sequences. *J Mol Graph*. **9**, 188-190.

P 83 Gibrat JF, Garnier J, Robson BJ (1987) Further developments of protein secondary structure prédiction using information theory. New parameters and consideration of residue pairs. *J. Mol. Biol.* **198**, 425-443.

P 84 Glemet E, Codani JJ (1997) LASSAP, a LArge Scale Sequence compArison Package. *Comput Applic Biosci*. **13**, 137-143.

P 85 Goldberg DE (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*.Addison-Wesley, New–York.

P 86 Goldman N, Thorne JL, Jones DT (1996) Using evolutionary trees in protein secondary structure prediction and other comparative sequence analysis. *J Mol Biol*. **263**, 196-208.

P 87 Goldman N, Thorne JL, Jones DT (1998) Assessing the impact of secondary structure and solvent accessibility on protein evolution. *Genetics*. **149**, 445-458.

P 88 Gouy M, Gautier C, Milleret F (1985), System analysis and nucleic acid sequence banks, *Biochimie* **67**, 433-436

P 89 Gracy J, Argos P (1998) Automated protein database classification. I. Integration of compositional similarity search, local similarity search and multiple sequence alignment. *Bioinformatics* **14**, 164-173

P 90 Gracy J, Argos P (1998) Automated protein database classification. II. Delineation of domain boudaries from sequence similarities. *Bioinformatics* **14**, 174-187

P 91 Grundy WN, Bailey TL, Elkan CP (1996) ParaMEME : a parallel implementation and a Web interface for the DNA and protein motif discovery tool. *Comput Applic Biosci*. **12**, 303-310.

P 92 Guan X, Du L (1998) Domain identification by clustering sequence alignment. *Bioinformatics*. **14**, 783-788.

P 93 Guermeur Y (1997) Combinaison de classifieurs statistiques. Application à la prédiction de la structure secondaire des protéines. Thèse de doctorat de l'Université Paris 6.

P 94 Guermeur Y, Geourjon C, Galinari P, Deléage G (1999) Improved performance in protein secondary structure prediction by inhomogeneous score combination. *Bioinformatics* (sous presse)

P 95 Guex N & Peitsch MC (1996), Swiss-Pdb Viewer : a fast and easy-to-use PDB viewer for Macintosh and PC. *PDB Quat Newslet*t, 77, 7

P 96 Gupta Sk, Kececioglu JD, Schäffer AA (1995) Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *J Comput Biol*. **2**, 459-472.

P 97 Harper R (1994) Access to DNA and protein databses in the Internet. *Curr Opin Biotechnol*. **5** 4-18.

P 98 Hartl FU (1996) Molecular chaperones in cellular protein folding. *Nature* **381**, 571-580.

P 99 Heinikoff S, Heinikoff JG, Alford WJ, Pietrokovsky S (1995) Automated construction and graphical presentaion of protein blocks from unaligned sequences. *Gene-COMBIS, Gene*. **163**, 17-26.

P 100 Henikoff JG, Henikoff S, Pietrokovski S (1999) New features of the blocks database servers. *Nucleic Acids Res.* **27**, 226-228.

P 101 Higgins D G, Sharp P M (1989), Fast and sensitive multiple alignments on a microcomputer. *Comput. Appl. Biosci.* **5**, 151-153.

P 102 Hobohm U, Sander C (1994) Enlarged representative set of protein structures. *Protein Science* **3**, 522-524

P 103 Hodges PE, McKee AHZ, Davis BP, Payne WE & Garrels JI (1999), The Yeast protein database (YPD) : a model the organization and presentation of genome-wide functionnal data, *Nucleic Acids Res.* **27**, 69-73

P 104 Hofmann K, Bucher P, Falquet L, Bairoch A (1999) The PROSITE database, its status in 1999. *Nucleic Acids Res.* **27**, 215-219.

P 105 Hofmann K, Bucher P, Falquet L, Bairoch A (1999), The PROSITE database, its status in 1999. *Nucleic. Acids Res.* **27**, 215-9

P 106 Hopp TP, Woods KR (1981) Prediction of protein antigenic determinants from amino acid sequences. *Proc. Natl. Acad. Sci. USA*, **78**, 3824-3828.

P 107 Huang X (1994) On global sequence alignment. *Comput Applic Biosci*. **10**, 227-235.

P 108 Hubbard TJ, Ailey B, Brenner SE, Murzin AG, Chothia C (1999) SCOP: a Structural Classification of Proteins database. *Nucleic Acids Res* **27**, 254-256

P 109 Jacob François. La Souris, la Mouche et l'Homme. Paris, Odile Jacob, 1997. 237p.

P 110 Janin J (1979) Surface and inside volumes in globular proteins. *Nature*, **277**, 491-492.

P 111 Jungfer K, Rodriguez-Tomé P (1998) Mapplet : a CORBA-based genome map viewer. *Bioinformatics*. **14**, 734-738.

P 112 Kabsch W, Sander C (1983) Dictionnary of protein secondary structure : pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers.* **22**, 2577-2637

P 113 Kanehisa M (1998) Databases of biological information. *Trends Guide to Bioinformatics*. 24-26.

P 114 Karp PD (1998) What we do not know about sequence analysis and sequence databases. *Bioinformatics* **14**, 753-754.

P 115 Karplus K, Barrett C, Hughey R (1998) Hidden Markov models for detecting remote protein homologies. *Bioinformatics* **14**, 846-856.

P 116 Karplus PA, Schulz GE (1985) Prediction of chain flexibility in proteins. *Naturwissenschaften*, **72**, 212-213.

P 117 Kernighan BW, Ritchie DM (1992) le langage C ANSI, 2e édition. Masson (Paris) et Prenctice Hall (London). 280p.

P 118 Kolakowski LF Jr. (1994), A G-protein-coupled receptor database. *Recept Channels*, **2**, 1-7

P 119 Kraemer ET, Ferrin TE (1998) Molecules to maps : tools for visualization and interaction in support of computational biology. *Bioinformatics*. **14**, 764-771.

P 120 Kyte J, Doolittle RF (1982) A simple method for displaying the hydropathic character of a protein. *J. Mol. Biol.*, **157**, 105-132.

P 121 Labesse G, Colloc'h N, Pothier J, Mornon JP (1997) P-SEA : a new efficient assignment of secondary structure from Cα trace of proteins. *Comput Applic Biosci.* **13**, 291-295

P 122 Laurie B, Laurie P (1997) Apache. Installation et Mise en Œuvre. O'Reilly, Paris. 296p.

P 123 Lefranc MP, Giudicelli V, Ginestoux C, Bodmer J, M ller W, Bontrop R, Lemaitre M, Malik A, Barbie V, Chaume D (1999) IMGT, the international ImMunoGeneTics database. *Nucleic Acids Res.* **27**, 209-212.

P 124 Lesk AM, Chothia C (1982) Evolution of proteins formed by beta-sheets. II. The core of the immunoglobulin domains. *J Mol Biol*. **160**, 325-342.

P 125 Levin JM, Pascarella S, Argos P, Garnier J (1993) Quantification of secondary structure prediction improvement using multiple alignment. *Prot. Eng.* **6**, 849-854

P 126 Levin JM, Robson B, Garnier J. (1986) An algorithm for secondary structure determination in proteins based on sequence similarity. *FEBS Lett.* **205**, 303-8

P 127 Levin, JM (1997) Exploring the limits of nearest neighbour secondary structure prediction. *Prot. Eng.* **7**, 771-776

P 128 Lewitter F (1998) Text-based database searching. *Trends Guide to Bioinformatics*. 3-5.

P 129 Liò P, Goldman N, Thorne JL, Jones DT (1998) PASSML : combining evolutionary inference and protein secondary structure prediction. *Bioinformatics*. **14**, 726-733.

P 130 Lipman DJ, Altschul SF, Kececioglu JD(1989) A tool for multiple sequence alignment. *Proc Natl Acad Sci USA.* **86**, 4412-4415.

P 131 McClure MA, Vasi TK, Fitch WM (1994) Comparative analysis of multiple protein-sequence alignment methods. *Mol Biol Evol*. **11**, 571-592.

P 132 McKusick VA (1994) Mendelian Inheritance in Man. Catalogs of Human Genes and Genetic Disorders. Baltimore: Johns Hopkins University Press (11th edition).

P 133 Médigue C, Rechenmann F, Danchin A, Viari A (1999) Imagene : an integrated computer environment for sequence annotation and analysis. *Bioinformatics*. **15**, 2-15.

P 134 Mewes HW, Heumann K, Kaps A, Mayer K, Pfeiffer F, Stocker S, Frishman D (1999) MIPS: a database for genomes and protein sequences. *Nucleic Acids Res.* **27**, 44-48.

P 135 Mizuguchi K, Go N (1995) Comparison of spatial arrangements of the secondary structure elements in proteins. *Protein Eng*. **8**, 353-362.

P 136 Morange Michel. La part des Gènes. Paris, Odile Jacob (Sciences), 1998. 222p.

P 137 Morgenstern B, Frech K, Dress A, Werner T (1998) DIALIGN : finding local similarities by multiple sequence alignment. *Bioinformatics*. **14**, 290-294.

P 138 Murzin AG, Brenner SE, Hubbard T, Chothia C (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* **247**, 536-540

P 139 Myers EW, Miller W (1988) Optimal alignments in linear space. *Comput Applic Biosci*. **4**, 11-17.

P 140 Needleman, Wunsch (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*. **48**, 443-453.

P 141 Netzer WJ, Hartl FU (1998) Protein folding in the cytosol : chaperonin-dependent and –independent mechanisms. *Trends Biochem Sci*. **23**, 68-73.

P 142 Nevill-Manning CG, Wu TD, Brutlag DL (1998) Highly specific protein sequence motifs for genome analysis. *Proc natl Acad Sci USA*. **95**, 5865-5871.

P 143 Notredame C, Higgins D (1996) SAGA: Sequence Alignmnet by Genetic Algorithm. *Nucleic Acids Res*. **24**, 1515-1524.

P 144 Notredame C, Holm L, Higgins DG (1998) COFFEE : an objective function for multiple sequence alignments. *Bioinformatics*. **14**, 407-422.

P 145 Ogata H, Goto S, Sato K, Fujibuchi W, Bono H, Kanehisa M (1999) KEGG : Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res*. **27**, 29-34.

P 146 Orengo CA, Michie AD, Jones S, Jones DT, Swindells MB, Thornton JM (1997) CATH--a hierarchic classification of protein domain structures. *Structure* **5**, 1093-1108

P 147 Orengo CA, Pearl FM, Bray JE, Todd AE, Martin AC, Lo Conte L, Thornton JM (1999) The CATH Database provides insights into protein structure/function relationships. *Nucleic Acids Res* **27**, 275-279

P 148 Orengo CA, Pearl FMG, Bray JE, Todd AE, Martin AC, Lo Conte L, Thornton JM (1999) ProClass protein family database. *Nucleic Acids Res.* **27**, 272-274.

P 149 Package GCG, Genetics Computer Group, Inc.

P 150 Parker JMR, Guo D, Hodges RS (1986) New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: correation of predicted surface residues with antigenicity and X-ray-derived accessible sites. *Biochemistry*, **25**, 5425-5432.

P 151 Parry-Smith DJ, Attwood TK (1992) ADSP--a new package for computational sequence analysis. *Comput Applic Biosci.* **8**, 451-459.

P 152 Pearson WR (1996) Effective protein sequence comparison. *Methods Enzymol*. **266**, 227-258.

P 153 Pearson WR, Lipman DJ (1988) Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA* **85**, 2444-2448.

P 154 Pierre Blanc (1993) PostScript. L'essentiel. Eyrolles, Paris. 187p.

P 155 Plauger PJ (1994) La bibliothèque C standard. Prenctice Hall, Englewood Cliffs. 518p.

P 156 Pocock MR, Hubbard T, Birney E (1998) SPEM : a parser for EMBL style flat file database entries. *Bioinformatics*. **14**, 823-824.

P 157 Qian N Sejnowski TJ (1988) Predicting the secondary structure of globular proteins using neural networks models. *J. Mol. Biol.* **196**, 697-709

P 158 Rao JK, Argos P (1986), A conformational preference parameter to predict helices in integral membrane proteins. *Biochem. Biophys. Acta.* **869**, 197-214

P 159 Rebhan M, Chalifa-Caspi V, Prilusky J, Lancet D (1998) GeneCards : a novel functional genomics compendium with automated data mining and query reformulation support. *Bioinformatics*. **14**, 656-664.

P 160 Rich-Text Format (RTF) version 1.0 Specification (1992) Microsoft Product Support Services. 34p.

P 161 Rodriguez-Tomé P (1998) The BioCatalog.*Bioinformatics* **14**, 469-470.

P 162 Rodriguez-Tomé P, Caterina D (1992) CEPH-Généthon.

P 163 Rognes T, Seeberg E (1998) SALSA : improved protein database searching by a new algorithm for assembly of sequence fragments into gapped alignments. *Bioinformatics* **14**, 839-845.

P 164 Rost B (1996). PHD : predicting one-dimensional protein structure by profile-based neural networks. *Methods Enzymol*. **266**, 525-539.

P 165 Rost B, O'Donoghue S (1997) Sisyphus and prediction of protein structure. *Comput Applic Biosci.* **13**, 345-356.

P 166 Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.* **232**, 584-599

P 167 Rost B, Sander C (1994) Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins* **19**, 55-72

P 168 Salamov AA, Solovyev VV (1997) Protein secondary structure prediction using local alignments. *J Mol Biol*. **268**, 31-36.

P 169 Sander C, Schneider R (1991). Database of homology-derived structures and the structural meaning of sequence alignments. *Proteins*. **9**, 56-68.

P 170 Sayle, RA Milner-White, EJ (1995), RASMOL : biomolecular graphics for all. *Trends Biochem Sci*. **19**, 258-260

P 171 Shpaer EG, Robinson M, Yee D, Candlin JD, Mines R, Hunkapiller T (1996) Sensitivity and selectivity in protein similarity searches : a comparison of Smith-Waterman in hardware to BLAST and FASTA. *Genomics* **38**, 179-191.

P 172 Smith RF, Smith TF (1992) Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependant gap penalties for use in comparative protein modelling. *Protein Eng.* **5**, 35-41.

P 173 Smith TF, Waterman MS (1981) Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195-197.

P 174 Srinivasarao GY, Yeh LSL, Marzec CR, Orcutt BC, Barker WC, Pfeiffer F (1999) Database of protein sequence alignments: PIR-ALN. *Nucleic Acids Res.* **27**, 284-285.

P 175 Stoesser G, Tuli MA, Lopez R, Sterk P (1999) the EMBL nucleotide sequence database. *Nucleic Acids Res*. **27**, 18-24.

P 176 Stoye J, Moulton V, Dress WM (1997) DCA : an efficient implementation of the divide-and-conquer approach to simultaneaous multiple sequence alignment. *Comput Applic Biosci*. **13**, 625-626.

P 177 Sweet, R (1986) Evolutionnary similarity among peptide segment is a basis for prediction of protein folding. *Biopolymers* **25**, 1565-1577

P 178 Swindell SR, Miller RR, Myers GSA (1996) Internet for the molecular biologist. Swindell SR, Miller RR, Myers GSA (eds.), Norfolk, England.

P 179 Taylor WR, Thornton JM (1983) Prediction of super-secondary structure in proteins. *Nature* **301**, 540-542

P 180 Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **22**, 4673-4680.

P 181 Thompson JD, Plewniak F, Poch O (1999) BAliBASE : a benchmark alignment database for the evaluation of multiple alignment programs *Bioinformatics* **15**, 87-8.

P 182 Thornton JM (1998) The future of Bioinformatics. *Trends Guide to Bioinformatics*. 30-31.

P 183 Vincens P, Buffat L, André C, Chevrolat JP, Boisvieux JF, Hazout S (1998) A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics*. **14**, 715-725.

P 184 Wall L, Christiansen T, Schwartz RL (1997) Programmation en Perl, 2e édition. O'Reilly International Thomson, Paris. 683p.

P 185 Welling GW, Weijer WJ, van der Zee R, Welling-Wester S (1985) Prediction of sequential antigenic regions in proteins. *FEBS Lett.*, **188**, 2, 215-218.

P 186 Wütrich K (1986) NMR of Proteins and Nucleic Acids. *Jon Wiley and Sons, New–York.*

P 187 Yan M, Lin ZS, Zhang CT (1998) A new Fourier transform approach for protein coding measure based on the format of the Z curve. *Bioinformatics*. **14**, 685-690.

P 188 Zhang X, Mesirov JP, Waltz DL (1992) Hybrid system for protein secondary structure prediction. *J. Mol. Biol.* **225**, 1049-1063

6.2 URL

URL 1 ACNUC. http://pbil.univ-lyon1.fr/search/query.html

URL 2 AnTheProt : Analyzing The Protein. http://www.ibcp.fr/ANTHEPROT/

URL 3 Apache : serveur Web, http://www.apache.org

URL 4 AT&T Bell Laboratories : http://www.belllabs.com

URL 5 BIOML :BIOpolymer Markup Langage, http://www.proteometrics.com/BIOML

URL 6 Bioperl :. http://bio.perl.org

URL 7 Blocks. http://www.blocks.fhcrc.org

URL 8 CATH : Class, Architecture, Topology or fold and Homologous family, http://www.biochem.ucl.ac.uk/bsm/cath

URL 9 CERN : Centre Européen pour la Recherche Nucléaire. http://www.cern.ch

URL 10 CORBA applications in Bioinformatics. http://industry.ebi.ac.uk/~corba

URL 11 Dbcat, a catalog of biological databases. http://www.infobiogen.fr/services/dbcat

URL 12 DCA Divide-and-Conquer Alignment. http://bibiserv.Tech-Fak.Uni-Bielefeld.de/dca

URL 13 DOMO : http://www.infobiogen.fr/~gracy/domo

URL 14 EBI : European Bioinformatic Institute, EMBL outstation. Hinxton hall, Cambridgeshire. http://www.ebi.ac.uk

URL 15 EMBL Nucleotide Sequence Database. http://www.ebi.ac.uk/embl.html

URL 16 EMBL : European Molecular Biology Laboratory, http://www.embl-heidelberg.de

URL 17 ExPASy :EXpert Protein Analysis SYstem, http://www.expasy.ch

URL 18 FreeBSD : UNIX libre, http://www.FreeBSD.org

URL 19 GCRDb : G protein Coupled Receptor Database. http://www.gcrdb.uthscsa.edu

URL 20 GenBank ®. http://www.ncbi.nlm.nih.gov

URL 21 GNU :Gnu Not Unix (Ensemble logiciel libre et complet de type UNIX) http://www.gnu.org

URL 22 IMGT : international ImMunoGeneTics database. http://imgt.cnusc.fr:8104

URL 23 InfoBioGen (GIS) : Groupement d'Intérêt Scientifique "Informatique appliquée à l'Etude des Biomolécules et des Génomes". http://www.infobiogen.fr

URL 24 IUPAC : International Union of Pure and Applied Chemistry. http://www.iupac.org

URL 25 JAVA : www.java.org, www.sun.com/java

URL 26 KEGG : Kyoto Encyclopedia of Genes and Genomes. http://www.genome.ad.jp/kegg

URL 27 Medline : http://www4.ncbi.nlm.nih.gov/Entrez/medline.html

URL 28 MIPS : Munich Information Center for Protein Sequence. http://pedant.mips.biochem.mpg.de

URL 29 MPSA : Multiple Protein Sequence Analysis. http://www.ibcp.fr/mpsa

URL 30 NCBI : National Center for Biotechnology Information, National Library of Medecine (NLM), National Institute of Health (NIH), Bethesda. http://ncbi.nlm.nih.gov

URL 31 netBSD : UNIX libre, http://www.netBSD.org

URL 32 OMG : Object Management Group (architecture distribuée CORBA). http://www.omg.com

URL 33 OMIM : Online Mendelian Inheritance in Man. http://www.ncbi.nlm.nih.gov/Omim

URL 34 openBSD : UNIX libre, http://www.openBSD.org

URL 35 PBIL Pôle Bioinformatique Lyonnais. http://pbil.univ-lyon1.fr

URL 36 PDB identity subset. ftp://ftp.embl-heidelberg.de/pub/databases/protein_extras/pdb_select

URL 37 PDB : Protein Data Bank. Brookhaven. http://www.pdb.bnl.gov jusqu'au 30 juin 1999. A partir du 1er juillet 1999 la PDB est hébérgée par le RCSB (Research Collaboratory for Structural Bioinformatics): http://www.rcsb.org

URL 38 PEDANT : Protein Extraction, description and Analysis Tool. http://pedant.mips.biochem.mpg.de

URL 39 PERL : http://www.perl.org, http://www.perl.com

URL 40 Pfam. http://www.sanger.ac.uk/Software/Pfam et http://www.cgr.ki.se/Pfam (Europe) ; http://pfam.wustl.edu (USA)

URL 41 PIR : Protein Information ressource. http://www–nbrf.georgetown.edu/pir
URL 42 PRINTS. http://www.biochem.ucl.ac.uk/bsm/dbbrowser
URL 43 PRODOM : database of domain families. http://protein.toulouse.inra.fr/prodom.html
URL 44 RasMol : Rastering Molecules. http://www.glaxowellcome.co.uk/software/
URL 45 SCOP : Structural Classification Of Proteins, http://scop.mrc-lmb.cam.ac.uk/scop
URL 46 SeqAnalRef : http://www.expasy.ch/seqanalref
URL 47 SGD : *Saccharomyces* Genome Database. http://genome-www.stanford.edu/Saccharomyces
URL 48 SIB : Swiss Institute of Bioinformatics. http://www.sib.ch
URL 49 SRS : Sequence Retrieval System. http://srs.ebi.ac.uk
URL 50 Swiss-Pdb Viewer. http://www.expasy.ch/spdbv/mainpage.html
URL 51 SWISS–PROT. http://www.expasy.ch/sprot
URL 52 Tcl/Tk :Tool Command Language/ ToolKit. http://www.neosoft.com
URL 53 The BioCatalog, release 6.0, 2 mars 1999. http://www.ebi.ac.uk/biocat
URL 54 The BioCatalog, submission form. http://www.ebi.ac.uk/biocat/biocat_form.html
URL 55 TIGR : The Institute for Genomic Research. Rockville, MD. http://www.tigr.org
URL 56 W3C : World Wide Web Consortium. http://www.w3.org
URL 57 XML : www.w3c.org/XML
URL 58 YPD : Yeast Protein Database. http://www.proteome.com/YPDhome.html

6.3 Références personnelles

6.3.1 Publications

P1 - Protein structure prediction. Implications for the biologist.
Deléage G., Blanchet C. and Geourjon C.
Biochimie, 1997, 79, 681-686

P2 - Logiciel MPSA et ressources bioinformatiques client-serveur Web dédiées à l'analyse de séquences de protéine.
Blanchet C.
Thèse de Doctorat de l'Université Claude Bernard - Lyon 1 soutenue le 31 mai 1999.

P3 - Molecular Model, Calcium Sensitivity, and Disease Specificity of a Conformational Thyroperoxidase B-cell Epitope.
Estienne V., Blanchet C., Niccoli-Sire P., Duthoit C., Durand-Gorde J.M., Geourjon C., Baty D., Carayon P et Ruf J.
J. Biol. Chem., 1999, 274, 35313-35317.

P4 - MPSA : Integrated System for Multiple Protein Sequence Analysis with client/server capabilities.
Blanchet C., Combet C., Geourjon C. et Deléage G.
Bioinformatics (sous presse).

P5 - NPSA: Network Protein Sequence Analysis
Combet C., Blanchet C., Geourjon C. et Deléage G.
Tibs (accepté).

6.3.2 Communications orales

C1 - Editeur d'alignements multiples de protéines couplés à des prédictions de structure secondaire.
Blanchet C., Geourjon C. et Deléage G.
23e Forum des Jeunes Chercheurs de la Société Française de Biochimie et Biologie Moléculaire. 2-5 juillet 1996. Poitiers

C2 - Prédiction de structure secondaire, un outil pour l'étude des relations structure-fonction des protéines.
Geourjon C., Blanchet C., Combet C. et Deléage G.
15e Colloque annuel de la Société Française de Biophysique. 22-25 septembre 1996

C3 - Utilisation de la fréquence des acides aminés pour la recherche de signatures fonctionelles.
Blanchet C., Geourjon C. et Deléage G.
Réunion de travail du Groupement De Recherche 1116 - Algorithmique, Modèles, Infographie (GDR-AMI). 2-3 décembre 1996. Marne la Vallée

C4 - Conception de domaines protéiques grâce au programme ANTHEPROT. Applications aux études RMN du domaine d'interaction avec l'ADN du facteur de choc thermique.
Deléage G., Geourjon C, Yang Y.S., Gagliardi D., Ladavière L., Montserret R., Blanchet C. et Penin F.
Congrès IMABIO. 16-17 décembre 1996. Strasbourg

C5 - Prédiction de structure secondaire des protéines: applications et perspectives.
Geourjon C., Blanchet C., Combet C. et Deléage G.
10e Journées du Groupe Graphique Moléculaire (GGM). 21-23 mai 1997. Dourdan

C6 - MPSA: Logiciel multi-plateforme d'analyse de séquences.
Blanchet C., Geourjon C. et Deléage G.
24e Forum des Jeunes Chercheurs de la Société Française de Biochimie et Biologie Moléculaire. 7-11 juillet 1997. Corte

C7 - Protein structure prediction and molecular biocomputing. Applications to protein "domain design".
Deléage G, Blanchet C., Combet C., Penin F. et Geourjon C.
2nd European Biophysics Congress. 13-17 Juillet 1997. Orléans

C8 - MPSA: une interface pour l'analyse de séquences de protéine sur le réseau Internet.
Blanchet C.
Table ronde sur la Bioinformatique.

26e Forum des Jeunes Chercheurs de la Société Française de Biochimie et Biologie Moléculaire.
21-22 juin 1999. Nice

6.3.3 Séminaires

S1 - Analyse de séquences en amont et en aval de la détermination de structures de protéines.
Démonstration du logiciel MPSA sur station Silicon Graphics.
7 novembre 1997. Institut de Biologie Structurale. Grenoble
Invitation de O. Diddeberg

S2 - Méthodes informatiques d'analyse de séquence de protéine.
Démonstration du logiciel MPSA sur Macintosh.
25 mars 1998. Institut de Biologie et Chimie des Protéines. Lyon

S3 - Environnement client-serveur pour l'analyse de séquences de protéines.
Démonstration du logiciel MPSA.
11es journées du Groupe de Graphisme et Modélisation Moléculaire. 10-12 mai 1999. Lyon

6.3.4 Posters

E1 - Délimitation de domaines protéiques à l'aide du programme ANTHEPROT. Applications pour l'étude structurale du domaine d'interaction avec l'ADN du facteur de choc thermique (HSF) de maïs par RMN.
Deléage G., Yang Y.S., Geourjon C., Montserret R., Blanchet C. et Penin F.
Journées Biologiques de Gerland. 20-21 janvier 1998. Lyon

E2 - MPSA: A software for Multiple Protein Sequence Analysis
Blanchet C., Geourjon C. and Deléage G.
European Conference: Computational chemistry and the living world from sequence to function. 20-24 avril 1998. Chambéry

E3 - MPSA: A software for Multiple Protein Sequence Analysis
Blanchet C., Geourjon C. and Deléage G.
2e Journée Scientifique de l'Ecole Doctorale Interdisciplinaire Science-Santé (EDISS). 20 mai 1998. Lyon

E4 - Outils client-serveur d'analyse de séquences protéiques
Blanchet C., Combet C., Geourjon C. and Deléage G.
9e Rencontres Régionales de la Recherches. 3 novembre 1998. Lyon

E5 - Le Pôle Bioinformatique Lyonnais - PBIL
Perrière G., Blanchet C., Duret L., Geourjon C., Gouy M. and Deléage G.
Journées de Biologie de Lyon. 21-22 janvier 1999. Lyon

E6 - MPSA-NPS@: Analyse de séquences de protéine en mode client-serveur
Blanchet C., Combet C., Geourjon C. and Deléage G.
Journées de Biologie de Lyon. 21-22 janvier 1999. Lyon

E7 - The Bioinformatic Pole of Lyon
Perrière G., Blanchet C., Duret L., Geourjon C., Thioulouse J., Combet C., Gouy M. and Deléage G.
Recomb'99. 11-14 avril 1999. Lyon
(Poster avec comité de lecture)

E8 - Protein Sequence Analysis software with Web facilities
Blanchet C., Geourjon C. and Deléage G.
Recomb'99. 11-14 avril 1999. Lyon
(Poster avec comité de lecture)

E9 - MPSA – Ressources bioinformatiques et réseaux pour l'analyse de séquences protéiques
Blanchet C., Combet C., Geourjon C. and Deléage G.
11es journées du GGMM. 10-12 mai 1999. Lyon

E10 - Remote query on biological Web servers from the protein sequence analysis software
Blanchet C., Combet C., Geourjon C. and Deléage G.
FEBS'99 - 26th Meeting of the Federation of European Biochemical Societies. 19-24 juin 1999. Nice

E11 - Structure of a human TPO peptide containing a conformational B-cell epitope specific for Hashimoto's thyroiditis
Estienne V., Blanchet C., Niccoli P., Duthoit C., Durand J., Baty D., Deléage G., Carayon P. et Ruf J.
FEBS'99 - 26th Meeting of the Federation of European Biochemical Societies. 19-24 juin 1999. Nice

E12 - Remote query on biological Web servers from the protein sequence analysis software
Blanchet C., Combet C., Geourjon C. and Deléage G.
26e Forum des Jeunes Chercheurs de la Société Française de Biochimie et Biologie Moléculaire.
21-22 juin 1999. Nice

6.3.5 Contrat de recherche

G1 - Prédictions de structures secondaires et bases de données de séquences.
Actions Concertées des Sciences du Vivant , ACC-SV 13 bioinformatique. 1995-1997. Responsable du projet: Gilbert Deléage. Montant alloué: 450 000 F TTC.

Annexes

Annexe A Logiciels extraits de BioCat

A1 Logiciels inventoriés

Rang	Nom	OS	Langage	Domaine	Ref. Biblio.
1	123D	-	-	1, 2, 3	1
2	ACCESS	UNIX, Vax	Pascal	4, 5	-
3	AE2	UNIX	C	6	-
4	ALIGN	UNIX, Vax	C	2	2, 3, 4
5	ALIGN - MBCRR	UNIX	C	2	-
6	ALSCRIPT	Dos, UNIX, Vax	ANSI-C	6	5
7	AMAS	Dec, Sgi, Sun, Vax	C	2, 3	6
8	AMPHI	Vax	FORTRAN	1, 5	7
9	AMPS	Sgi, Sun, Vax	Fortran 77 and C	2	8, 9
10	ANTHEPROT	Dos, UNIX, Win	fortran, C, Visual Basic	1, 2, 6, 7, 8, 9, 10, 11	10, 11, 12, 13, 14, 15
11	ANTHEPROT Web page	-	-	1, 7, 9, 10, 12	16, 17
12	ARIADNE	Sun, Vax	Any Common Lisp System	9	-
13	ASSET	Sgi, Sun	C	2	18
14	BANDS	-	C	2	19
15	BCM Search Launcher	Mac, UNIX	Perl (on Unix), AppleScript (on Macintosh)	1, 2, 3, 4, 6, 7, 8, 9, 10, 13, 14, 15, 16, 17	20
16	BEAUTY	Mac, UNIX	Perl (on Unix), AppleScript (on Macintosh)	3, 4, 7, 15, 16, 17, 18	21
17	BESTSCOR	-	C	2	-
18	BIOSCAN	Dec, Sun	-	2	22, 23
19	BLAST	Ibm, Mac, Sgi, SunIX	C	2	24
20	BLAST and FASTA scripts	-	-	2	-
21	BLAST-SHELL	Vax	-	2, 5	-
22	BLAST3	UNIX	C	2	24
23	BLASTINC	Ibm, Mac, Sgi, SunIX	C	2	24
24	BLASTLIB	Ibm, Mac, Sgi, SunIX	C	2	24
25	BLASTN	Ibm, Mac, Sgi, SunIX	C	2	24
26	BLASTP	Ibm, Mac, Sgi, SunIX	C	2	24
27	BLASTPAT and FASTPAT	-	-	2, 3, 7	-
28	BLASTVMS	Vax	-	2	24
29	BLASTX	-	-	2	24
30	BLAZE	-	-	2	-
31	BLIMPS	-	C	2, 15	25
32	BMB	UNIX	C	2	26
33	BOB	Dec, Sun	C (+ X11R5 and Motif)	10	-

Rang	Nom	OS	Langage	Domaine	Ref. Biblio.
34	BOXSHADE	-	Pascal	6	-
35	BTAB	UNIX	C, yacc, lex	10	27
36	Bielefeld University Bioinformatics Server	-	English	2, 3, 7, 18, 19, 20	-
37	BioMotif	UNIX	C	9	-
38	BlueGene	Win	-	1, 4, 7, 15, 21, 22, 23	-
39	CAIC	Mac	-	17, 20	28
40	CBLAST	UNIX	C, perl	4, 7	29
41	CBLAST	UNIX	C, perl	4, 7	29
42	CDB (Chemical Database)	-	C	4, 24	-
43	CLEVER	-	-	4	-
44	CLUSTAL V	UNIX, Vax	C	2, 20	30
45	CLUSTAL W	Dos, Mac, UNIX, Vax	C	2, 20	31
46	CONSENSUS	Dec, Sgi, Sun	C	2	32, 33
47	CREGEX	Dos, UNIX, Vax	C	9	34
48	CloneIt Online	-	-	3, 9, 14, 18	35
49	ClustalX	Mac, UNIX, Win	C	2, 10, 20	36
50	DARWIN	-	-	2	-
51	DASHER3	UNIX	C	2	-
52	DBWATCHER	Sgi, SunIX	ANSI C	2, 4	-
53	DCSE	Sgi	C,Tcl/Tk	6	37
54	DFBLAST	-	C	2	38
55	DNA Stacks	Mac	HyperTalk, C	6, 7, 9, 10, 13, 18, 20, 22, 23, 25	39
56	DOTPLOT	UNIX	C, X11 (Xview)	2	40
57	DROSOPOSON	Sun	LE_LISP	15, 16, 17, 19, 26	41
58	DTASK11S	Dec, Sgi, Sun	C	2	-
59	DYNAMIC	UNIX	C	2	-
60	Dali server	UNIX, Vax	Perl,Fortran	4, 8, 27, 28	42
61	Dali server	UNIX	Perl,Fortran	4, 8, 27	42
62	EASYMENU	Vax	C	5, 7	-
63	EGCG	Dec, Vax	-	5, 7	43, 44
64	EMBL-Search	Win	-	4	-
65	EMBL-Search	Mac	-	4	45
66	ENTREZ	Ibm, Mac, PC, Sun, Win	C	4	-
67	FASTA	-	-	2	46, 47
68	FASTEMBL	Vax	DCL	2	-
69	FAStRNA	UNIX	C	3, 4, 29	48
70	FILTER	Sun	FORTRAN C	2	49, 50
71	FSAP	Dos, Ibm, Sun	C Pascal	2	-
72	FSSP	-	Perl	3, 8, 10	51
73	GAP	Dos, PC, Sun	C	2	52
74	GCG Wisconsin Package	Ibm, Sgi, Sun, Vax	Fortran, C	5, 7	-

Rang	Nom	OS	Langage	Domaine	Ref. Biblio.
75	GCGDBASE	-	FORTRAN C	5, 7	-
76	GCGEMBL	-	FORTRAN C	5, 7	-
77	GCGMENU	-	FORTRAN C	5, 7	53
78	GCGQUICK	-	FORTRAN C	5, 7	-
79	GCGSHELLS1	Vax	FORTRAN C	5, 7	-
80	GCGSHELLS2	Vax	DCL, VAX FORTRAN C	5, 7	-
81	GDB Human Genome Data Base	-	-	3, 4, 15, 25	54, 55
82	GDE	Dec, Sgi, SunIX, Win	C	7, 30	56
83	GENAL	UNIX	C	2	-
84	GIBBS	-	The data structures used in this program are part of a package of object oriented C code for molecular biological applications developed by Andrew F. Neuwald.	2	57
85	GRAILSHELLS	Vax	FORTRAN C DCL	5, 31	-
86	GenQuest	Dec, Mac, Sun, Win	-	2	-
87	GeneDoc	Win	-	6, 7, 20	-
88	GeneMan	Mac, Win	-	4	-
89	HCDSearch	UNIX	Pascal	2, 4, 18	58
90	HMMER	Dec, Sgi, SunIX	ANSI C	2	-
91	HOMOCHART	UNIX	C	6	-
92	INTERFACE	Vax	Pascal C	5	-
93	INTERVALS & POINTS	UNIX	C	2	59
94	ISSC	Dec	FORTRAN C UIS or DEC-Windows		2
60, 61					
95	JOY	UNIX	Fortran, C, Perl	3, 6, 7, 8, 17	62
96	KabatMan	-	-	3, 4, 15	-
97	LALIGN	-	-	2	-
98	LALNVIEW	Mac, PC, UNIX	ANSI C, SUIT	6, 7	-
99	LAV	-	C	2	63
100	LCP	Sun	C	2	64
101	LFASTA	-	-	2	65
102	LOCAL	UNIX	C	2	66
103	MACAW	Mac, PC, Win	-	2, 6	57, 67, 68
104	MALI and PRALI	SunIX, Vax	-	2	69
105	MALIGNED	-	-	6	-
106	MAP	Sun	C	2	52
107	MASE	UNIX	C	6	70
108	MBLKP and MBLKN	Dos, PC, Sun	C	2	67, 71
109	MEMFILE	-	C	2	-
110	MENUGCG	UNIX	csh script	5, 7	-

Rang	Nom	OS	Langage	Domaine	Ref. Biblio.
111	MOBILITY	Vax	FORTRAN	5	-
112	MOTIF	Vax	C	2	72
113	MOTIFS	Dec, Sgi, Sun, Vax	C, Fortran	9	-
114	MSA	UNIX	C	2, 3	73, 74, 75, 76, 77, 78
115	MSE	Vax	C	6	-
116	MView	UNIX	perl	10	79
117	MacPattern	Mac	-	9	80
118	Map123d	-	-	1, 3, 6, 7, 8, 11	-
119	Matrix Search	UNIX	C	4, 7, 15	81
120	Miropeats	Sgi, SunIX	C-shell, ANSI-C	2, 6, 7, 13, 18	82, 83
121	Motif Master	Dos	-	9	-
122	NCBISHELLS	Vax	FORTRAN C DCL	5, 32	-
123	NOCUT	-	C	5, 14	-
124	NSEQTOOL	Sun	C SunView	6	84
125	OMIM	-	Text file in the FTP site	3, 4	-
126	OSA	Sgi, SunIX	GPC Pascal	7, 7, 16, 17, 20, 33	85
127	OVERSEER	UNIX, Vax	Pascal	4	86
128	PAM	-	-	2	-
129	PATMAT	Dos, UNIX	-	4, 9	87
130	PATTERN	Dos, Ibm, UNIX	F77 Fortran	9	13
131	PATTERN	Vax	C	9	88
132	PDBTOGCG	Vax	FORTRAN	5, 13	-
133	PIMA	UNIX	C	2	89, 90
134	PIP	Dec, UNIX, Vax	-	9	91
135	PLALIGN	-	-	2	-
136	PLFASTA - PCLFASTA	-	-	2	-
137	PLSEARCH	UNIX	C	2, 9	92
138	PRATT	UNIX, Vax	ANSI C	7, 9	93, 94
139	PRDF	-	-	2	-
140	PRESSDB	-	-	2	-
141	PRO-EXPLORE	Sgi	-	6, 9, 11	-
142	PROFILEWEIGHT	UNIX, Vax	C	5	95
143	PROMOT	Sgi, Sun	-	9	96
144	PROSEARCH	UNIX	C, awk	9	97
145	PROSITE	Dos	-	9	-
146	PROSITE	Ibm, UNIX	C	9	-
147	PROSITE	Mac	-	9, 15	-
148	PROSITEC	Vax	Pascal	5, 13	-
149	PROTOMAT	Dos, Sun	-	9	98
150	PRSS	-	-	2	-
151	PSB (Prosite.doc Browser)	-	C	4	-
152	PSQ	Vax	-	4	-

Rang	Nom	OS	Langage	Domaine	Ref. Biblio.
153	PYTHIA	Sun	UNIX C-shell scripts	2	-
154	PairWise and SearchWise	Sgi, Sun, Vax	C	2, 4, 7	-
155	ProDom PROTEIN DOMAIN WWW SERVER	-	-	2, 3, 4, 7, 15	99
156	ProMSED (Protein Multiple Sequence Editor)	Win	C++	2, 6, 7	100
157	QUELSITE	Vax	FORTRAN	4	101
158	QUEST	Sun, Vax	-	9	-
159	RASA 2.2	Mac, PC	-	1, 7, 9, 16, 17, 20	102, 103, 104, 105, 106
160	RDP Recursive Dynamic Programming	-	-	1, 2, 3	107
161	RNABOB	UNIX	C	4, 29	108, 109
162	RNCBI2	UNIX	tcl/tk	4	-
163	ROBUST	-	C	2	110
164	RSS	-	-	2	-
165	RTHEORY	UNIX	C	2	111, 112
166	RepeatMasker	UNIX	Perl, C	3, 4, 7, 27	-
167	SAGA	UNIX	C	2, 7	113
168	SAM	UNIX	-	2	114, 115, 116, 117, 118, 119
169	SCRUTINEER	UNIX, Vax	Pascal	9	120, 121
170	SCUO	UNIX, Vax	FORTRAN	5	122
171	SEALS: A System for Easy Analysis of Lots of Sequences	UNIX	Perl	4, 7, 9, 13, 15, 18, 20, 34	123
172	SEAVIEW	Dec, Ibm, Sgi, Sun	-	6, 20	-
173	SEQF	UNIX	FORTRAN	4	-
174	SEQ_VIS	UNIX	C	9	124
175	SETDB	-	-	2	-
176	SIGNPT	Vax	FORTRAN	9	125
177	SIM	UNIX, Vax	C	2	126
178	SIM2	Mac, UNIX, Win	C, NCBI software toolkit	2	127
179	SIMPLE34	Sun, Vax	FORTRAN77	2	128
180	SP2FASTA	-	-	2	-
181	SQUIRREL	Vax	Pascal	5, 7	129
182	SRS	UNIX, Vax	-	4	130, 131
183	SSEARCH	-	-	2	-
184	STAMP	Sgi, Sun	C and fortran	2	132, 133
185	STATALIGN	UNIX	C	2, 20	134
186	STATSEARCH	UNIX, Vax	FORTRAN	4	135
187	Seq-Eudora-Blast	Mac	English	4	-
188	SeqVu	Mac	-	6	-
189	Sequence Logos	-	-	6	136

Rang	Nom	OS	Langage	Domaine	Ref. Biblio.
190	Sequin	Dec, Mac, PC, Sgi, SunIX, Win	English	6, 7, 10, 13, 18, 22, 23, 35	-
191	ShadyBox	UNIX	-	6	-
192	TBLASTN	-	C	2	-
193	TBOB	UNIX	Perl	10	-
194	TDALIGN	Dos, PC, SunIX, Vax	C and FORTRAN	2	137
195	TDF2GCG	Vax	FORTRAN C	5, 13	-
196	TFASTA	-	-	2	-
197	TNB	-	-	2	138
198	TREEALIGN	UNIX, Vax	C	2, 20	139
199	TULLA	-	-	2	-
200	TargetFinder	-	-	3, 4, 9, 34	140, 141
201	The USC Sequence Alignment Package	-	C	2, 3, 7, 9, 16, 18	-
202	ToPLign Toolbox for Protein aLignments	-	-	2, 3, 6, 7, 10	142
203	Visual BLAST (and Visual FASTA)	Win	-	6, 7	143
204	WHAT-IF	Dos, UNIX	FORTRAN, C	2, 4, 7, 8, 11, 15	144
205	WORDUP	UNIX	C	9	145
206	WPDB	PC, Win	C++	4	146
207	WU-BLASTN	UNIX	C	2, 4	147
208	WU-BLASTP	UNIX	C	2, 4	147
209	WU-BLASTX	UNIX	C	2, 4	148
210	WU-TBLASTN	UNIX	C	2, 4	147
211	WU-TBLASTX	UNIX	C	2, 4	147
212	WWW-Query	UNIX	C, Fortran	3, 4, 7, 36	149
213	WWW2GCG	Dec, Sgi, SunIX	C,Perl, JavaScript, Java	5, 7	150
214	Webin PESTFIND	-	Perl, JavaScript ANSI C	3, 7, 9, 35	-
215	XALIGN	Sgi, Sun	Ansi C	2	151
216	XBLAST	UNIX	C	2	152
217	XNU	UNIX	C	2	153
218	XYLEM	SunIX	Pascal, C, csh.	4	154
219	nrdb90 K-Estimator 4.3	PC	Perl Visual Basic 5	4, 7, 18, 37	155

219 logiciels retenus sur 580

A2 Champs d'application

(1) Structure prediction
(2) Alignment Search software
(3) WWW server
(4) Searching databases
(5) GCG tools
(6) Alignment editing and display
(7) Protein sequence analysis
(8) Protein structure analysis
(9) Pattern Identification

(10) Alignment browser
(11) Molecular modelling and graphics
(12) WWW_server
(13) Sequence format conversion tools
(14) Restriction maps
(15) Database and analysis
(16) Statistical significance
(17) Comparative analysis
(18) Sequence tools
(19) Genetic tools
(20) Phylogeny
(21) Neural Networks
(22) Sequence display
(23) Sequence editor
(24) Chemistry
(25) Genome Mapping Databases
(26) Population Genetics
(27) e-mail Server ANALYSIS
(28) WWW Server
(29) RNA analysis
(30) Alignment editing and dipslay
(31) e-mail Server tools
(32) e-mail Server RETRIEVAL and ANALYSIS
(33) Epidemiology
(34) Genome Analysis
(35) Sequence data submissions
(36) Multivariate analysis
(37) Molecular Evolution

A3 Références bibliographiques

(1) Alexandrov N., Nussinov R., Zimmer R.; "Fast Protein Fold Recognition via Sequence to Structure Alignment and Contact Capacity Potentials"; Proceedings of the First Pacific Symposium on Biocomputing, 53-72 (1996)

(2) Pearson W.R., Lipman D.J.; "Improved Tools for Biological Sequence Comparison"; Proc. Natl. Acad. Sci. USA 85:2444-2448(1988).

(3) Pearson W.R.; "Rapid and Sensitive Sequence Comparison with FASTP and FASTA"; Methods in Enzymology 183:63-98(1990).

(4) Myers E., Miller W.; "Optimal Alignments in Linear Space"; Comput. Appl. Biosci. 4:11-17(1988).

(5) Barton G.J.; "ALSCRIPT a tool to format multiple sequence alignments."; Protein Eng. 6:37-40(1993).

(6) Livingstone C.D., Barton G.J; "Protein Sequence alignments: a strategy for the hierarchical analysis of residue conservation"; Comput. Appl. Biosci. 9:745-756(1993).

(7) Jahnig F.; "Structure predictions of membrane proteins are not that bad."; Trends Biochem. Sci. 15:93-95(1990).

(8) Barton G.J.; "Protein Multiple Sequence Alignment and Flexible Pattern Matching"; Meth. Enzymol. 183:403-428(1990).

(9) Barton G.J., Sternberg M.J.E; "A Strategy for the Rapid Multiple Alignment of Protein sequences: Confidence Levels From Tertiary Structure Comparisons"; J. Mol. Biol. 198:327-337(1987).

(10) Geourjon C., Deleage G., Roux B.; "ANTHEPROT : An interactive graphic software for analyzing protein structures from sequences."; J. Mol. Graph. 9:188-190(1991).

(11) Deleage G., Clerc F.C., Roux B., Gautheron D.C.; "ANTHEPROT: a package for protein sequence analysis using a microcomputer."; Comput. Appl. Biosci. 4:351-356(1988).

(12) Deleage G., Clerc F.F., Roux B.; "ANTHEPROT: IBM PC and Apple MacIntosh versions."; Comput. Appl. Biosci. 5:159-160(1989).

(13) Geourjon C., Deleage G.; "Interactive and graphic coupling between multiple alignments, secondary structure predictions and motif/pattern scanning into proteins."; Comput. Appl. Biosci. 9:87-91(1993).

(14) Deleage G., Geourjon C.; "An interactive graphic program for calculating secondary structures content of proteins from circular dichroism spectrum."; Comput. Appl. Biosci. 9:197-199(1993).

(15) Deleage G., Geourjon C.; "ANTHEPROT 2.0: A three-dimensional module fully coupled with protein sequence analysis methods."; J. Mol. Graph. 13:209-212(1995).

(16) Geourjon C., Deleage G.; "SOPM: a self-optimised meyhod for protein secondary struture prediction."; Protein Eng. 7:157-164(1994)

(17) Geourjon C., Deleage G.; "SOPMA:significant improvements in protein secondary structure prediction by consensus prediction from multiple alignments."; Comput. Appl. Biosci. 11(6):681-684(1995).

(18) Neuwald A.F., Green P.; "Detecting patterns in protein sequences"; J. Mol. Biol. 239:698-712(1994).

(19) Chao K.-M., Pearson W.R., Miller W.; "Aligning two sequences within a specified diagonal band."; Comput. Appl. Biosci. 8:481-487(1992).

(20) Smith R.F., Wiese B.A., Wojzynski M.K., Davison D.B., Worley K.C.; "BCM Search Launcher--An integrated interface to molecular biology database search and analysis services available on the World-Wide Web."; Submitted.

(21) Worley K.C., Wiese B.A., Smith R.F.; "BEAUTY: An enhanced BLAST-based search tool that integrates multiple biological information resources into sequence similarity search results."; Genome Research 5:173-184 (1995).

(22) White C.T., Singh R.K., Reintjes P.B., Lampe J., Erickson B.W., Dettloff W.D., Chi V.L., Altschul S.F.; "BioSCAN : A VLSI-Based System for Biosequence Analysis,"; 1991 IEEE International Conference on Computer Design: VLSI in Computers & Processors, IEEE Computer Society Press, Los Alamitos, CA, pp.504-509 (1991).

(23) Singh R.K., Tell S.G., White C.T., Hoffman D., Chi V.L., Erickson B.W.; "A scalable systolic multiprocessor system for analysis of biological sequences."; (In) Research on integrated systems. Proceedings of the 1993 symposium; Borriello G., Ebeling C., Eds., pp168-182, MIT Press, (1993).

(24) Altschul S.F., Gish W., Miller W., Myers E.W., Lipman D.J.; "Basic local alignment search tool."; J. Mol. Biol. 215:403-410(1990).

(25) Henikoff S., Henikoff J.G.: "Automated assembly of protein blocks for database searching"; Nucleic Acids Res. 19(23):6565-6572 (1991).

(26) Chao K.-M., Hardison R.C., Miller W.; "Constrained sequence alignment"; Bull. Math. Biol. 55(3):503-524(1993).

(27) Dubnick M.; "Btab - A Blast Output Parser."; Comput. Appl. Biosci. 8:601-602(1992).

(28) Purvis A., Rambaut A.; (1995) "Comparative analysis by independent contrasts (CAIC): an Apple Macintosh application for analysing comparative data."; Comp. Appl. in Biosci. 11(3):247-251(1995).

(29) Miller, G.S. and Fuchs R.; "Post-processing of BLAST results using databases of clustered sequences."; Comput. Appl. Biosci. 13:81-87(1997).

(30) Higgins D.G., Bleasby A.J., Fuchs R.; "CLUSTAL V: improved software for multiple sequence alignment."; Comput. Appl. Biosci. 8:189-191(1992).

(31) Thompson J.D., Higgins D.G., Gibson T.J.; "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice."; Nucleic Acids Res. 22:4673-4680(1994).

(32) Stormo G.D., Hartzell G.W; "Identifying protein-binding sites from unaligned DNA fragments"; Proc. Natl. Acad. Sci. U.S.A 86:1183-1187(1989).

(33) Hertz G.Z., Hartzell G.W., Stormo G.D.; "Identification of consensus patterns in unaligned DNA sequences known to be functionally related."; Comput. Appl. Biosci. 6:81-92(1990).

(34) None (if necessary cite as unpublished method)

(35) Lindenbaum P.; "CloneIt: finding cloning strategies, in-frame deletions and frameshifts."; Bioinformatics 14(5):465-466(1998).

(36) Thompson J.D, Gibson T.J, Plewniak F, Jeanmougin F, Higgins D.G.; "The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools."; Nucleic Acids Res. 25:4876-4882(1997).

(37) De Rijk P., De Wachter R.; "DCSE v2.54, an interactive tool for sequence alignment and secondary structure research."; Comput. Appl. Biosci. 9:735-740(1993).

(38) Claverie J.-M., States D.J.; "Information enhancement methods for large scale sequence analysis."; Comput. Chem. 17:191-201(1993).

(39) Eernisse, D. J.; "DNA Translator and Aligner: HyperCard utilities to aid phylogenetic analysis of molecules."; Comput. Appl. Biosci. 8:177-184(1992).

(40) Trelles-Salazar O., Zapata E.L., Dopazo J., Coulson A.F.W., Carazo J-M.; "An image-processing approach to dotplots: an X-Window-based program for interactive analysis of dotplots derived from sequence and structural data."; Comp. Appl. Biosci. 11(3):301-308(1995).

(41) Hoogland C., Biemont C.; "DROSOPOSON: a knowledge base on chromosomal localization of transposable element insertions in Drosophila."; Compu. Appl. Biosci. 13:61-68(1997).

(42) Holm L., Sander C.; "Protein structure comparison by alignment of distance matrices."; J. Mol. Biol. 233:123-138(1993).

(43) Gibson T.J., Rice P.M., Thompson J.D., Heringa J.; "KH domains within the FMR1 sequence suggest that fragile X syndrome stems from a defect in RNA metabolism."; Trends Biochem. Sci. 18:331-333(1993).

(44) Musacchio A., Gibson T., Rice P., Thompson J., Saraste M.; "The PH domain: a common piece in the structural patchwork of signalling proteins."; Trends Biochem. Sci. 18:343-348(1993).

(45) Fuchs R., Stoehr P.; "EMBL-Search - A CD-ROM based database query system."; Comput. Appl. Biosci. 9:71-77(1993).

(46) Pearson W.R., Lipman D.J.; "Improved Tools for Biological Sequence Analysis."; Proc. Natl. Acad. Sci. U.S.A 85:2444-2448(1988).

(47) Pearson W.R.; "Rapid and Sensitive Sequence Comparison with FASTP and FASTA."; Meth. Enzymol. 183:63-98(1990).

(48) El Mabrouk N., Lisacek F. "Very fast identification of RNA lotifs in genomic DNA. Application to tRNA search in the yeast genome." J.Mol.Biol.264:46-55 (1996).

(49) Vingron M., Argos P.; "Determination of reliable regions in protein sequence alignments."; Protein Eng. 3:565-569(1990).

(50) Vingron M., Argos P.; "Motif recognition and alignment for many sequences by comparison of dot-matrices."; J. Mol. Biol. 218:33-43(1991).

(51) Holm L., Sander C.; The FSSP database of structurally aligned protein fold families. Nucleic Acids Res. 22:3600-3609(1994).

(52) Huang X.; "On global sequence alignment."; Comput. Appl. Biosci. 10:227-235(1994).

(53) Gartmann C.J., Grob U.; "A menu-shell for the GCG programs."; Comput. Appl. Biosci. 7:457-460(1991).

(54) Fasman, K.H., Letovsky, S.I., Cottingham, R.W., Kingsbury, D.T.; Nucleic Acids Res. in press (1996).

(55) Fasman, K.H., Cuticchia, A.J., Kingsbury, D.T.; "The GDB human genome data base anno 1994."; Nucleic Acids Res. 22:3462-3469 (1994).

(56) Smith S.W., Overbeek R., Woese C.R., Gilbert W., Gillevet P.M.; "The Genetic data environment and expandable GUI for multiple sequence analysis."; Comput. Appl. Biosci. 10(6):671-675(1994).

(57) Lawrence C.E., Altschul S.F., Boguski M.S., Liu J.S., Neuwald A.F., Wootton J.C.; "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment"; Science 262:208-214(1993).

(58) Cocea L.; "Search the Human cDNA Database at TIGT."; Comput. Appl. Biosci. 13(2):201-204(1997).

(59) Miller W.; "Building multiple alignments from pairwise alignments."; Comput. Appl. Biosci. 9:169-176(1993).

(60) Argos P.; "A sensitive procedure to compare amino acid sequences."; J. Mol. Biol. 193:385-396(1987).

(61) Rechid R., Vingron M., Argos P.; "A new interactive protein sequence alignment program and comparison of its results with widely used algorithms."; Comput. Appl. Biosci. 5:107-113(1989).

(62) Mizuguchi K., Deane C.M., Johnson M.S., Blundell T.L., Overington J.P.; "JOY: protein sequence-structure representation and analysis."; Bioinformatics 14:617-623(1998).

(63) Schwartz S., Miller W., Yang C.-M., Hardison R.C; "Software tools for analyzing pairwise alignments of long sequences."; Nucleic Acids Res. 19:4663-4667(1991).

(64) Huang X.; "An algorithm for identifying regions of a DNA sequence that satisfy a content requirement"; Comput. Appl. Biosci. 10:219-225(1994).

(65) Chao K.-M., Pearson W.R., Miller W; "Aligning two sequences within a specified diagonal band."; Comput. Appl. Biosci. 8:481-487(1992).

(66) Smith T.F., Waterman M.S.; "Identification of common molecular subsequences."; J. Mol. Biol. 147:195-197(1981).

(67) Schuler G.D., Altschul S.F., Lipman D.J.; "A Workbench for Multiple Alignment Construction and Analysis."; Proteins 9:180-191(1991).

(68) Karlin S., Altschul S.F.; "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes."; Proc. Natl. Acad. Sci. U.S.A. 87:2264-2268(1990).

(69) Vingron M., Argos P.; "A fast and sensitive multiple sequence alignment algorithm."; Comput. Appl. Biosci. 5(2):115-121(1989).

(70) Faulkner D.V., Jurka J.; "Multiple aligned sequence editor (MASE)."; Trends Biochem. Sci. 13:321-322(1988).

(71) Karlin S., Altschul S.F.; "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes."; Proc. Natl. Acad. Sci. U.S.A. 87:2264-2268(1990).

(72) Cockwell K.Y., Giles I.G.; "Software tools for motif and pattern scanning: program descriptions including a universal sequence reading algorithm."; Comput. Appl. Biosci. 5:227-232(1989).

(73) Lipman D.J., Altschul S.F., Kececioglu J.D.; "A tool for multiple sequence alignment."; Proc. Natl. Acad. Sci. U.S.A. 86:4412-4415(1989).

(74) Carrillo H., Lipman D.J.; "The Multiple Sequence Alignment Problem in Biology."; SIAM J. Appl. Math. 48:1073-1082(1988).

(75) Altschul S.F., Lipman D.J.; "Trees, Stars, and Multiple Biological Sequence Alignment."; SIAM J. Appl. Math. 49:197-209(1989).

(76) Altschul S.F.; "Gap Costs for Multiple Sequence Alignment."; J. Theor. Biol. 138:297-309(1989).

(77) Altschul S.F., Carroll R.J., Lipman D.J.; "Weights for Data Related by a Tree."; J. Mol. Biol. 207:647-653(1989).

(78) Altschul S.F.; "Leaf Pairs and Tree Dissections."; SIAM J. Discrete Math. 2:293-299(1989).

(79) Brown, N.P., Leroy, C., Sander, C. "MView: A Web compatible database search or multiple alignment viewer."; Bioinformatics (1998).

(80) Fuchs R.; "MacPattern: protein pattern searching on the Apple MacIntosh."; Comput. Appl. Biosci. 7:105-106(1991).

(81) Chen Q.K., Hertz G.Z., Stormo D.G.; "MATRIX SEARCH 1.0: a computer program that scans DNA sequences for transcriptional elements using a database of weight matrices."; Comput. Appl. Biosci. 11(5):563-566(1995).

(82) Parsons J.D.; "Miropeats: graphical DNA sequence comparisons."; Comput. Appl. Biosci. 11(6):615-619(1995).

(83) Parsons J.D.; "Improved tools for DNA comparison and clustering."; Comput. Appl. Biosci. 11(6):603-613(1995).

(84) Schnobel R.; "Integrated displays of aligned amino acid sequences and protein structures."; Comput. Appl. Biosci. 7:341-346(1991).

(85) Martin M.J., Gonzalez-Candelas F., Sobrino F., Dopazo J.; "A method for determining the position and size of optimal sequence regions for phylogenetic analysis."; J. Mol. Evol. 41:1128-1138(1995).

(86) Sibbald P.R., Sommerfeldt H., Argos P.; "Overseer: a nucleotide sequence searching tool."; Comput. Appl. Biosci. 8:45-48(1992).

(87) Wallace J.C., Henikoff S.; "PATMAT: a searching and extraction program for sequence, pattern and block queries and databases."; Comput. Appl. Biosci. 8:249-254(1992).

(88) Cockwell K.Y., Giles I.G.; "Software tools for motif and pattern scanning: program descriptions including a universal sequence reading algorithm."; Comput. Appl. Biosci. 5:227-232(1989).

(89) Smith R.F., Smith T.S.; "Automatic generation of primary sequence patterns from sets of related protein sequences."; Proc. Natl. Acad. Sci. U.S.A. 87:118-122(1990).

(90) Smith R.F., Smith T.S.; "Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for comparitive protein modelling."; Protein Eng. 5:35-41 (1992).

(91) Staden R.; "Screening protein and nucleic acid sequences against libraries of patterns."; "DNA Seq. 1:369-374(1991).

(92) Smith R.F., Smith T.S.; "Automatic generation of primary sequence patterns from sets of related protein sequences."; Proc. Natl. Acad. Sci. U.S.A. 87:118-122(1990).

(93) Jonassen I., Collins J. F., Higgins D. G.; "Finding flexible patterns in unaligned protein sequences."; Protein Science (1995) 4:1587-159;

(94) Jonassen I.; "Efficient discovery of conserved patterns using a pattern graph."; subm. to CABIOS February 1997

(95) Thompson J.D., Higgins D.G., Gibson T.J.; "Improved sensitivity of profile searches through the use of sequence weights and gap excision."; Comput. Appl. Biosci. 10:19-29(1994).

(96) Sternberg M.J.E.; "PROMOT: A FORTRAN program to scan protein sequences against a library of known motifs."; Comput. Appl. Biosci. 7:257-260(1991).

(97) Kolakowski L.F. Jr., Leunissen J.A.M., Smith J.E.; "ProSearch: fast searching of protein sequences with regular expression patterns related to protein structure and function."; Biotechniques 13:919-921(1992).

(98) Henikoff S., Henikoff J.; "Automated assembly of protein blocks for database searching."; Nucleic Acids Res. 19:6565-6572(1991).

(99) Sonnhammer E.L.L., Kahn D.; "Modular arrangement of proteins as inferred from analysis of homology."; Protein Sci. 3:482-492(1994).

(100) Frolov A., Eroshkin A.; "ProMSED: Protein Multiple Sequence Editor for Windows 3.x/95."; In preparation

(101) Dessen P., Fondrat C., Valencien C., Mugnier C.; "BISANCE: A French service for access to biomolecular sequence databases."; Comput. Appl. Biosci. 6:355-356(1990).

(102) Lyons-Weiler, J., Hoelzer, G.A. and Tausch R.J.; "Relative Apparent Synapomorphy Analysis (RASA) I: the statistical measurement of phylogenetic signal."; Molecular Biology and Evolution 13:749-757(1996).

(103) Lyons-Weiler, J. and Milinkovitch, M.C.; "A phylogenetic approach to the problem of differential lineage sorting."; Molecular Biology and Evolution 14:968-975(1997).

(104) Lyons-Weiler, J. and Hoelzer, G.A.; "Escaping from the Felsenstein Zone by detecting long branches in phylogenetic data."; Molecular Phylogenetics and Evolution 8:375-384(1997).

(105) Lyons-Weiler, J. and Hoelzer, G.A.; "Optimal outgroup analysis."; Biological Journal of the Linnean Society 64:493(1998).

(106) Milinkovitch, M.C. and Lyons-Weiler, J.; "Finding optimal outgroup topologies and convexities when the choice of outgroups is not obvious."; Molecular Phylogenetics and Evolution 9:348-357(1998).

(107) Thiele, R., Zimmer, R., Lengauer, T.; "Recursive Dynamic Programming for Adaptive Sequence and Structure Alignment"; Proc. of the Third International Conference on Intelligent Systems for Molecular Biology, 384-392(1995).

(108) Gautheret D., Major F. and Cedergren R.; "Pattern searching/alignment with RNA primary and secondary structures: an effective descriptor for tRNA."; Comp. Apps. Biosci. 6:325-331 (1990).

(109) Eddy S.R.; "RNABOB: a program to search for RNA secondary structure motifs in sequence databases."; Unpublished.

(110) Chao K.-M., Hardison R.C., Miller W.; "Locating well-conserved regions within a pairwise alignment."; Comput. Appl. Biosci. 9:387-396(1993).

(111) Allison L., Wallace C.S., Yee C.N.; "Finite-state models in the alignment of macromolecules."; J. Mol. Evol. 35:77-89(1992).

(112) Yee C.N., Allison L.; "Reconstruction of strings past."; Comp. Appl. Biosci. 9(1):1-7(1993).

(113) Notredame C., Higgins D.; "SAGA: Sequence Alignmnet by Genetic Algorithm."; Nucleic Acids Res. 24(8):1515-1524(1996).

(114) Haussler D., Krogh A., Mian S., Sjolander K.; "Protein Modeling Using Hidden Markov Models."; (In) C.I.S. Technical Report UCSC-CRL-92-23, University of California at Santa Cruz, 1992.

(115) Krogh A., Brown M., Mian I.S., Sjolander K., Haussler D.; "Hidden Markov Models in Computational Biology: Applications to Protein Modeling."; J. Mol. Biol. 235:1501-1531(1994).

(116) Baldi P., Chauvin Y., Hunkapiller T.; "Hidden Markov Models of Biological Primary Sequence Information."; Proc. Natl. Acad. Sci. U.S.A. 91:1059-1063(1994).

(117) Rabiner R.L.; "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition."; Proc. IEEE 77:257-286(1989).

(118) Eddy S.R., Mitchison G., Durbin R.; "Maximum Discrimination Hidden Markov Models of Sequence Consensus."; J. Computational Biology, in press.

(119) Eddy S.R.; "Multiple Alignment Using Hidden Markov Models."; to be presented at ISMB '95, Cambridge, UK.

(120) Sibbald P.R., Argos P.; "Scrutineer: a computer program that flexibly seeks and describes motifs and profiles in protein sequence databases."; Comput. Appl. Biosci. 6:279-288(1990).

(121) Sibbald P.R., Sommerfeldt H., Argos P.; "Automated protein sequence pattern handling and PROSITE searching."; Comput. Appl. Biosci. 7:535-536(1991).

(122) Zhou J., Moody M.E., Johns S.J., Kleinhofs A.; "An Informational Index for Measuring Codon Usage Bias.";

(123) Walker DR, Koonin EV; "SEALS: A System for Easy Analysis of Lots of Sequences"; ISMB 5:333-339(1997).

(124) Smith T.F., Srinivasan A., Schochetman G., Marcus M., Myers G.; "The phylogenetic history of immunodeficiency viruses."; Nature 333:573-575(1988).

(125) Dessen P., Fondrat C., Valencien C., Mugnier C. "BISANCE: A French service for access to biomolecular sequence databases."; Comput. Appl. Biosci. 6:355-356(1990).

(126) Huang X., Hardison R.C., Miller W.; "A space-efficient algorithm for local similarities."; Comput. Appl. Biosci. 6:373-381(1990).

(127) Chao K-M, Zhang J., Ostell J., Miller W.; "A local alignment tool for very long DNA sequences."; Comput. Appl. Bios. 11(2):147-153(1995).

(128) Hancock J.M., Armstrong J.S.; "SIMPLE34: an improved and enhanced implementation for VAX and Sun computers of the SIMPLE algorithm for analysis of clustered repetitive motifs in nucleotide sequences."; Comput. Appl. Biosci. 10:67-70(1994).

(129) Gartmann C.J., Grob U.; "SQUIRREL: Sequence QUery, Information Retrieval and REporting Library. A program package for analyzing signals in nucleic acid sequences for the VAX."; Nucleic Acids Res. 19(21):6033-6040(1991).

(130) Etzold T., Argos P.; "SRS - an indexing and retrieval tool for flat file data libraries."; Comput. Appl. Biosci. 9:49-57(1993).

(131) Etzold T., Argos P.; "Transforming a set of biological flat file libraries to a fast access network."; Comput. Appl. Biosci. 9:59-64(1993).

(132) Russell R.B., Barton G.J.; "Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels"; Proteins 14:309-323(1992).

(133) Russell R.B., Barton G.J.; "An SH2-SH3 domain hybrid."; Nature 364:765(1993).

(134) Thorne J.L, Kishino H.; MBE 9:1148-1162(1992).

(135) Mott R.F., Kirkwood T.B.L.; "STATSEARCH: a GCG-compatible program for assessing statistical significance during DNA and protein databank searches."; Comput. Appl. Biosci. 6:293-295(1990).

(136) Schneider T.D., Stephens R.M.; "Sequence logos: a new way to display consensus sequences."; Nucleic Acids Res. 18:6097-6100(1990).

(137) Davison D.B., Thompson K.H.; "A non-metric sequence alignment algorithm."; Bull. Math. Biol. 46:579-590(1984).

(138) Boguski M.S., Hardison R.C., Schwartz S., Miller W.; "Analysis of conserved domains and sequence motifs in cellular regulatory proteins and locus control regions using new software tools for multiple alignment and visualization."; New Biol. 4:247-260(1992).

(139) Hein J.; "Unified approach to alignment and phylogenies."; Meth. Enzymol. 183:626-645(1990).

(140) Lavorgna et al. Trends in Genetics 14:375-376(1998).

(141) Lavorgna et al, 1998. Bioinformatics, in press

(142) Mevissen H, Thiele, R., Zimmer, R., Lengauer, T.; "The ToPLign software environment - Toolbox for Protein ALignment"; Bioinformatics'94, IMB, Jena (1994).

(143) Durand P., Canard L., Mornon J.P. "Visual BLAST and Visual FASTA : graphic workbenches for interactive analysis of full BLAST and FASTA outputs under Microsoft Windows 95/NT"; Comput. Appl. Biosci. 13(4):407-413(1997).

(144) Vriend G.; "WHAT IF: a molecular modelling and drug design program."; J. Mol. Graph. 8:52-56(1990).

(145) Pesole, G., Prunella, N., Liuni, S., Attimonelli, M. and Saccone C.; "WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences."; Nucleic Acids Res. 20:2871-2875 (1992).

(146) Shindyalov I.N., Bourne P.E.; -; J. App. Cryst. 28(6):847-852(1995).

(147) Altschul S.F. and Gish W.; "Local alignment statistics."; Methods in Enzymology 266:460-80(1996).

(148) Altschul S.F. and Gish W.; "Local alignment statistics."; Methods in Enzymology 266:460-80 (1996).

(149) Perriere, G. Thioulouse, J.; "On-line tools for sequence retrieval and multivariate statistics in molecular biology."; Comput. Applic. Biosci. 12:63-69(1996).

(150) Colet M., Herzog R.; "WWW2GCG, a web interface to the GCG biological sequences analysis software"; Comput. & Graphics 20:445-450(1996).

(151) Wishart D.S., Boyko R.F., Sykes B.D.; "Constrained multiple sequences alignment using XALIGN."; Comput. Appl. Biosci. 10(6):687-688(1994).

(152) Claverie J.-M., States D.J.; "Information enhancement methods for large scale sequence analysis."; Comput. Chem. 17:191-201(1993).

(153) Fristensky, B.; "Feature expressions: creating and manipulating sequence datasets."; Nucleic Acids Res. 21:5997-6003(1993).

(154) Holm, L. and Sander, C.; "Removing near-neighbour redundancy from large protein sequence collections."; Bioinformatics 14:423-429(1998). Comeron, J.M.; A method for estimating the numbers of synonymous and nonsynonymous substitutions per site; J. Mol. Evol. 41:1152-1159 (1995).

Annexe B Script de compilation de MPSA

```
#
# Makefile for building MPSA (LCP products)
#

######################
# usefull pathways
######################

# binaries directory
DIRBIN = /usr/local/bin/

# IBCP LCP top directory
IBCPLIBP = $(HOME)/lib
IBCPINCL = $(HOME)/include
#IBCPLIBP = /usr/local/lib
#IBCPINCL = /usr/local/include/ibcp
IBCPLIBS = -libcp

# Vibrant top directory
VIBLIBP = /usr/local/lib
VIBINCL = /usr/local/include/ncbi
VIBLIBS = -lvibrant -lncbi -lnetcli

# system library directory
SYSLIBP = /usr/lib
SYSINCL = /usr/include
SYSLIBS = -lXm -lXmu -lXt -lX11 -lm

######################
# compiler variables
######################

# c compiler and linker
CC = cc
ANSI =
#ANSI = -ansi

# Application Binary Interfaces( ABI)
#ABI = -n32 # par defaut implique -mips3
ABI =

# main target name
MAINTARGET = dpsa

# developpment flag
DEV = -DDEVELOP

#for Vibrant widget test
#TEST = -DTEST
TEST =

# debugger
# add below for cc version lower than 3.19
DEBUG = -g -DIBCPuse
# add below for cc version upper or equal to 3.19
#DEBUG = -D

# profiler
#PROF = -p
PROF =

# optimization
#OPT = -O
OPT =

# operating system :  LCP_UNIX, LCP_MAC, LCP_PC pour MPSA
#                     WIN_X (defini auto si WIN_MOTIF est def)
#                     , WIN_MAC, WIN_MSWIN pour Vibrant
OS = -DLCP_UNIX -DWIN_MOTIF

# compile flags
CFLAGS  = $(ABI) -c $(ANSI) $(OPT) $(DEBUG) $(PROF) $(DEV) $(TEST) $(OS) \
        -I$(VIBINCL) -I$(IBCPINCL) -I$(SYSINCL)
```

```
# linker flags
# LFLAGS = -s
LDFLAGS =

OBJ =  psa_alignment.o psa_alimodif.o psa_browser.o \
       psa_color.o psa_custom.o psa_display.o \
       psa_dotplot.o psa_file.o psa_font.o \
       psa_graph.o psa_help.o psa_info.o psa_main.o \
       psa_methods.o psa_mouse.o psa_overview.o \
       psa_physchemgui.o \
       psa_preferences.o psa_print.o \
       psa_scroller.o psa_secdetview.o psa_secpred.o \
       psa_secsumbox.o psa_secstruct.o \
       psa_tools.o psa_util.o psa_winmanag.o \
       psa_www.o

SRC =  psa_alignment.c psa_alimodif.c psa_browser.c \
       psa_color.c psa_custom.c psa_display.c \
       psa_dotplot.c psa_file.c psa_font.c \
       psa_graph.c psa_help.c psa_info.c psa_main.c \
       psa_methods.c psa_mouse.c psa_overview.c \
       psa_physchemgui.c \
       psa_preferences.c psa_print.c \
       psa_scroller.c psa_secdetview.c psa_secpred.c \
       psa_secsumbox.c psa_secstruct.c \
       psa_tools.c psa_util.c psa_winmanag.c \
       psa_www.c

##########
# targets
##########
#              -B static \           # pour sun, pour utiliser que les lib statiques
$(MAINTARGET): $(OBJ)
               $(CC) $(OBJ) -o $@ \
               $(ABI) \
               -L$(IBCPLIBP) $(IBCPLIBS) \
               -L$(VIBLIBP) $(VIBLIBS) \
               -L$(SYSLIBP) $(SYSLIBS)
               chmod 755 $@
#              mv $@ ../debug/

# add below for cc version lower than 3.19
#.c.o :$(SRC)
#      $(CC) $(CFLAGS)  $?

# install  file
install:
       mv $(MAINTARGET) $(DIRBIN)$(MAINTARGET)

# clean object file
clean:
       rm -f $(OBJ)
```

Annexe C Les principales variables de l'implémentation de MPSA

C1 Variable Lcp_Structure

```
typedef struct {
       int    type; /* sub-type of primary: csPrimAli or csPrimBseq */

       int    nbsec;
       Lcp_PtrStruct sec; /* first sec struct if current is primary */
       Lcp_PtrStruct seccons; /* not take in account in nbsec, local secondary consensus */

       WindoW win_seqfile; /* where the seq file is diplayed */
       WindoW win_physchem; /* view of physico-chemical profiles */
       WindoW win_aacompo; /* view of aa composition */
       } Lcp_PrimaryStructExtraData;

typedef struct {
       int  type; /* cf libibcp: secondary.h csSecXxxxx */

       /* sec state data */
       int nbstate; /* nb d'etat de struct sec */
       char* usedstate; /* cf secstructorder: [1,csSecStateNb] */
       double* stperc; /* percent for each sec state in the struct */

       /* pred calcul parameters */
       char* predparam;

       /* scores detailles */
       char scoreav; /* =1 si les scores sont disponibles (i.e. non mail) */
       WindoW win_score; /* fenetre des scores */
       } Lcp_StructExtraDataSecondary;

typedef struct {
       int  type; /* cf physchem.h de libibcp */
       char* seq; /* phys-chem sequence */
       double* score; /* score array */
       int compwin; /* scoring window size */
       int compwinhalf; /* scoring window size */
       WindoW win_score; /* fenetre des scores */
       } Lcp_StructExtraDataPhysChem;

typedef struct lcpstruct {
       char   type; /* csPrimaryStruct, csSecondaryStruct, csPattern,... */
       char   selected; /* =1 if struct selected */
       char   numero[csStructNumeroLeng+1]; /* number of the struct */
       char*  name; /* name of the struct, don't free if !flg_freename */
       int flg_freename; /* =1 when freeing name is allowed */
       char*  seq; /* sequence, pointer on extra->seq, don't free it */
       char*  path; /* complete pathway of the struct file */
       char*  info; /* comments about the structure */
       long   absrow; /* numero absolu de ligne dans l'ali/bseq */
       long   woutgaplen; /* longueur sans les gap */

       Lcp_PtrStruct  relprim; /* relative primary: itself if primary, primary to reference if
other struct type */
       Lcp_PtrStruct  next; /* next structure: prim, sec, motif, */
       Lcp_PtrStruct  prev; /* previous structure: prim, sec, motif, */
       Lcp_PtrStruct  nextprim; /* next primary structure */

       /* extra data */
       Lcp_PrimaryStructExtraData* exprim; /* extra data for a primary */
       Lcp_StructExtraDataSecondary* exsec; /* extra data for a secondary */
       void* extra; /* extra data */
       void (*free_extra) (void*);

       /* display */
       Lcp_PtrAliOrBseq ali; /* pointeur sur l'ali la contenant */
       Lcp_PtrAliDisplay disp; /* pointeur sur le display utilise */
       void (*drawelmt) (Lcp_PtrStruct);
       void (*drawline) (Lcp_PtrStruct);

       /* memory */
       void (*free) (Lcp_PtrStruct);
       Lcp_PtrStruct (*realloc) (Lcp_PtrStruct, long);
```

```
        /* printing */
        double (*getprheight) (Lcp_PtrStruct, void*);
        int (*writePSline2pr) (Lcp_PtrStruct, void*, FILE*);
        int (*writeRTFline2pr) (Lcp_PtrStruct, void*, FILE*);

        /* leng and query about aa */
        long (*getleng) (Lcp_PtrStruct, long, long);
        long (*getlengwoutgap) (Lcp_PtrStruct, long, long);
        long (*getposwoutgap) (Lcp_PtrStruct, long);
        long (*getposwgap) (Lcp_PtrStruct, long);
        char* (*getseqwoutgap) (Lcp_PtrStruct);

        /* misc */
        void (*fitonprim) (Lcp_PtrStruct, long, long, int);
        long (*getstructnumero) (Lcp_PtrStruct);
        } Lcp_Struct;
```

C2 Variable Lcp_AliOrBseq

```
typedef struct lcpaliorbseq {
        /* type */
        char type; /* csBaseSeq, csAlignment */
        Lcp_AlignmentExtraData* exali; /* extra data for an alignment */
        Lcp_BseqExtraData* exbseq; /* extra data for a sequence base */

        /* gal data */
        char* name; /* nom de l'ali/bseq */
        char* path; /* chemin complet de l'ali/bseq */
        int flg_need2besaved; /* >0 when ali/bseq need to be saved */
        int flg_closeaftersaved; /* >0 when ali/bseq need to be close after having been saved
*/

        /* structure data */
        char gap4prim;
        char gap4bseq;
        char gap4sec;
        char gap4pc;
        char gap4aamost;
        Lcp_Struct* list; /* liste chainee des structures */
        Lcp_Struct** map; /* cartes des structures */

        /* ali/bseq data size */
        void (*free) (Lcp_PtrAliOrBseq);
        long nbcol; /* nb de colonnes de l'alignement */
        long nbrow; /* nb de lignes de l'alignement */
        long nbprimary; /* nombre de structures primaires */
        long nbprimsel; /* nombre de structures primaires selectionnees */
        long nbsecondary; /* nombre de structures secondaires */
        long nbsecsel; /* nombre de structures secondaires selectionnees */

        /* mouse */
        Lcp_AASelection* aasel; /* selection d'aa a la souris */
        WindoW win_regsel; /* region selection by sepcifying rows and cols */

        /* display */
        int flg_nowavail; /* =1 when ali/bseq is completely read */
        int flg_scrolling; /* =1 si scroller en action */
        void (*display) (Lcp_PtrAliOrBseq);
        void (*displayhscrol) (Lcp_PtrAliOrBseq);
        Lcp_PtrAliDisplay disp; /* display ou s'affiche l'ali */

        } Lcp_AliOrBseq;
```

C3 Variable Lcp_AliDisplay

```
typedef struct lcpalidisplay {
        Lcp_PtrAliOrBseq ali; /* pointeur sur l'alignement */

        /* alignment drawing */
        PaneL pan_ali; /* panel ou s'affiche l'ali */
        BaR bar_h; /* alignment HORI scroll bar */
        BaR bar_v; /* alignment VERTI scroll bar */
        int    flg_noredraw4ali; /* =1 s'il ne faut pas redessiner l'ali */
        int    flg_needredraw4ali; /* =1 s'il faut redessiner l'ali */
        int flg_showseqnumb; /* =1 if showing sequence number */
        int flg_showseqname; /* =1 if showing sequence name */
```

```
        int flg_graphalicoupling; /* =1 if alignment display is linked to graph mouse */
        long nbcol;
        long nbrow;
        long numwid; /* seq nb length in char */
        long namewid; /* seq name length in char */
        long idwid; /* seq nb and name reserved space length in char */
        long poshscrol; /* position de l'ascenseur horizontal */
        long posvscrol; /* position de l'ascenseur vertical */
        long curcolbeg, curcolend; /* de 0 a disp->nbcol-1 */
        long currow; /* de 0 a disp->nbrow-1 */
        int* row2pix; /* numero de ligne vers sa position en pixels */
        int* col2pix; /* numero de colonne vers sa position en pixels */

        RecT rect_panali; /* espace d'affichage max dispo */
        RecT rect_ali; /* rectangle d'affichage de l'ali, i.e. un bloc */
        RecT rect_names; /* rectangle d'affichage des noms de struct */
        RecT rect_scale; /* rectangle d'affichage de l'echelle */
        RecT rect_cons; /* rectangle d'affichage du consensus */
        RecT rect_primcons; /* rectangle d'affichage de la seq cons des struct prim */
        RecT rect_seccons; /* rectangle d'affichage de la seq cons des struct sec */

        int     margin_aboveali; /* marge au-dessus de l'ali en pixels, i.e. hauteur de
l'echelle */
        int     margin_belowali; /* marge en dessous de l'ali en pixels, i.e. hauteur des
consensus */

        /* secondary structures */
        Lcp_SecData* secgd; /* global sec data */

        /* phys-chem prediction */
        int physchem_nb; /* physico-chemical meth nb */
        double *physchem_thresh; /* physico-chemical threshold for the display in ali/bseq */

        /* mouse */
        char flag_mousedragfromali; /* =1 si la souris est partie de l'ali */
        char flag_mousedragfromnames; /* =1 si la souris est partie des noms de struct */
        char mse_flgmvstruct; /* =1 if moving struct with mouse */
        long mse_fstrow; /* first row from where drag occurs */
        long mse_currow; /* cur row of dragging */
        PoinT mousefirstpt;
        PoinT mousecurpt;

        /* font */
        Lcp_PoliceCaract* font_ali;
        Lcp_PoliceCaract* font_scale;
        Lcp_PoliceCaract* font_info;

        /* color */
        Lcp_ColorMap*  cmap_gal; /* colormap generale */
        Lcp_ColorMap*  cmap_prim; /* colormap des structures primaires */
        Lcp_ColorMap*  cmap_sec;/* colormap des structures secondaires */
        Lcp_ColorMap*  cmap_picons; /* colormap pour le cons d'identite des struct prim */
        Lcp_Gradation* grad_ali; /* degrade pour l'alignement */
        Lcp_Gradation* grad_picons; /* degrade pour le cons d'identite des struct prim */
        WindoW win_colorsense; /* fenetre des significations de couleurs */
        PaneL  pan_colorsense; /* panel des significations de couleurs */

        /* gestion des choix autorises */
        ChoicE chce_color; /* colormode des struct primaires */
        int    primcolormode; /* colormode des struct primaires */
        char*  itemsav; /* necess a la gestion des objets de choix: AVailibility */
        char*  itemsav_last; /* stock le itemsav une mise a jour selective */
        void** items; /* tableau des objets de choix: menus, boutons,... */
        int    nbitems;

        /* remembering */
        Lcp_Remember* remember;

        /* object for the displaying window */
        WindoW win_main; /* fenetre d'affichage */
        int    lostwid, losthei; /* portion de la fen perdue pour l'ali */
        int flg_cmdline; /* TRUE if command line mode */
        int flg_silent; /* TRUE if silent mode: no message prompt to stdout */
        DoC    doc_message; /* boite des messages */
        WindoW win_mess; /* window for displaying software message */
        PrompT prp_fileload; /* affichage du nom de l'ali/bseq etudie */
        PrompT prp_status; /* boite des statuts */
```

```
/* customization windows */
WindoW win_custcmapprim;
WindoW win_custcmapsec;
WindoW win_custcgradprim;
WindoW win_custcgradconsprim;
WindoW win_custpathway;

/* usefull windows */
WindoW uwin_clustalparam; /* window for Clustal Parameter */
WindoW uwin_clustalhelp; /* window for Clustal help */
WindoW uwin_multalinparam; /* window for Multalin Parameter */
WindoW uwin_pattparam; /* window for Pattern Parameter */
WindoW uwin_pattprotparam; /* window for Pattinprot Parameter */
WindoW uwin_fastaparam; /* window for Fasta Parameter */
WindoW uwin_blastparam; /* window for Fasta Parameter */

WindoW win_seqbrow; /* sequences browser window */
WindoW win_seqcomments; /* sequences comments window */
WindoW win_find; /* window for searching string */
WindoW win_secselect; /* window for selecting some sec type */

WindoW win_dotplot; /* dot plot window */
} Lcp_AliDisplay;
```

Annexe D Routines d'affichage d'une séquence en couleur dans MPSA

```
/*****************************************************************************
*
*       DISPLAY_PrimLineFromBseq (Lcp_Struct* curprim)
*              affichage d'une struct prim issue d'une base de seq
*
*****************************************************************************/
static void DISPLAY_PrimLineFromBseq (Lcp_Struct* curprim)
{
        Lcp_AliDisplay* disp=curprim->disp;
        int lineleng=disp->curcolend-disp->curcolbeg;
        long abscol=disp->poshscrol+disp->curcolbeg;
        long ii;

        /* surlignage des aminoacides selectionnes */
        Highlighter (disp);

        /* preparation de la ligne en couleur 0 */
        sprintf (*seqtemp, "%-*.*s", lineleng, lineleng, curprim->seq+abscol);

        /* ecriture de la ligne */
        WRITE_Line (disp->pan_ali
                        , *(disp->col2pix+disp->curcolbeg)
                        , *(disp->row2pix+disp->currow)
                        , *seqtemp, *(disp->cmap_gal->val+csGalCmapBseq));

        /* nettoyage */
        memset (*seqtemp, ' ', lineleng+1);

        return;
}



/*****************************************************************************
*
*   DISPLAY_PrimLineInFamilycolor (Lcp_Struct* curprim)
*       affichage d'une struct prim dans le mode familycolor
*
*****************************************************************************/
static void DISPLAY_PrimLineInFamilycolor (Lcp_Struct* curprim)
{
  int ii;
  Lcp_AliDisplay* disp=curprim->disp;
  int lineleng=disp->curcolend-disp->curcolbeg;
  long abscol=disp->poshscrol+disp->curcolbeg;

  /* surlignage des aminoacides selectionnes */
  Highlighter (disp);

  /* preparation des lignes en couleur par famille */
  for (ii=0;ii<lineleng;ii++)
    *(*(seqtemp+(*((*(colorfam.all+colorfam.cur)).atofamcolor+(*(curprim-
>seq+ii+abscol)))))+ii)
        = *(curprim->seq+abscol+ii);

  /* affichage des differentes couleurs */
  for (ii=0;ii<(*(colorfam.all+colorfam.cur)).nbcolor;ii++) {
    *(*(seqtemp+ii)+lineleng) = '\0';
    WRITE_Line (disp->pan_ali
                , *(disp->col2pix+disp->curcolbeg)
                , *(disp->row2pix+disp->currow)
                , *(seqtemp+ii)
                , *(disp->cmap_prim-
>val+(*((*(colorfam.all+colorfam.cur)).famcolortoabscolor+ii))));
    *(*(seqtemp+ii)+lineleng) = ' ';  /* ne jamais supprimer sinon probleme lors
                                du dessin d'une ligne plus grande que la precedente */
  }

  /* nettoyage */
  for (ii=0;ii<lineleng;ii++)
    *(*(seqtemp+(*((*(colorfam.all+colorfam.cur)).atofamcolor+(*(curprim-
>seq+ii+abscol)))))+ii) = ' ';
```

```
  return;
}



/*****************************************************************************
*
*   DISPLAY_PrimLineInSimilaritycolor (Lcp_Struct* curprim)
*       affichage d'une struct prim dans le mode familycolor
*
*****************************************************************************/
static void DISPLAY_PrimLineInSimilaritycolor (Lcp_Struct* curprim)
{
  int ii;
  Lcp_AliDisplay* disp=curprim->disp;
  int lineleng=disp->curcolend-disp->curcolbeg;
  long abscol=disp->poshscrol+disp->curcolbeg;

  /* surlignage des aminoacides selectionnes */
  Highlighter (disp);

  /* preparation de la ligne */
  for (ii=0;ii<lineleng;ii++)
    *(*(seqtemp+(*(colorsim.columntocolor+abscol+ii)))+ii) = *(curprim->seq+abscol+ii);

  /* affichage de la ligne */
  for (ii=0;ii<colorsim.nbcolor;ii++)
  {
    *(*(seqtemp+ii)+lineleng) = '\0';
    WRITE_Line (disp->pan_ali
               , *(disp->col2pix+disp->curcolbeg)
               , *(disp->row2pix+disp->currow)
               , *(seqtemp+ii)
               , *(disp->cmap_prim->val+(*(colorsim.colorused+ii))));
    *(*(seqtemp+ii)+lineleng) = ' '; /* ne jamais supprimer sinon probleme lors
                                du dessin d'une ligne plus grande que la precedente */
  }

  /* nettoyage */
  for (ii=0;ii<lineleng;ii++)
    *(*(seqtemp+(*(colorsim.columntocolor+abscol+ii)))+ii) = ' ';

  return;
}



/*****************************************************************************
*
*       DISPLAY_SecLine (Lcp_Struct* cursec)
*               affichage d'une struct secondaire
*
*****************************************************************************/
static void DISPLAY_SecLine (Lcp_Struct* cursec)
{
        Lcp_AliDisplay* disp=cursec->disp;
        char *pchar;
        int lineleng=disp->curcolend-disp->curcolbeg;
        int ii;
        long abscol = disp->poshscrol+disp->curcolbeg;

        /* surlignage des positions selectionnes */
        Highlighter (disp);

        /* preparation de la ligne */
        pchar = cursec->seq+abscol;
        for (ii=0;ii<lineleng;ii++, pchar++)
                *(*(seqtemp+(*(disp->secgd->atostorder+*pchar)))+ii) = *pchar;

        /* affichage de la ligne */
        pchar = cursec->exsec->usedstate;
        for (ii=0;ii<cursec->exsec->nbstate;ii++, pchar++) {
                *(*(seqtemp+*pchar)+lineleng) = '\0';
                WRITE_Line (disp->pan_ali
                        , *(disp->col2pix+disp->curcolbeg)
                        , *(disp->row2pix+disp->currow)
                        , *(seqtemp+*pchar)
```

```
                        , *(disp->cmap_sec->val+*pchar));
                *(*(seqtemp+*pchar)+lineleng) = ' '; /* DON'T TOUCH the 'lineleng', this line
is here to remove '\0' */
                        /* ne jamais supprimer sinon probleme lors du dessin d'une ligne plus
grande que la precedente */
        }

        /* nettoyage */
        pchar = cursec->seq+abscol;
        for (ii=0;ii<lineleng;ii++, pchar++)
                *(*(seqtemp+(*(disp->secgd->atostorder+*pchar)))+ii) = ' ';

        return;
}



/*****************************************************************************
*
*   DISPLAY_PhysChemLine (Lcp_Struct* cursec)
*       affichage d'un phys-chem
*
*****************************************************************************/
static void DISPLAY_PhysChemLine (Lcp_Struct* curpc)
{
  Lcp_StructExtraDataPhysChem* expc=(Lcp_StructExtraDataPhysChem*)curpc->extra;
  Lcp_AliDisplay* disp=curpc->disp;
  int lineleng=disp->curcolend-disp->curcolbeg;
  int ii;
  long abscol=disp->poshscrol+disp->curcolbeg;

  /* surlignage des positions selectionnes */
  Highlighter (disp);

  /* preparation de la ligne en couleur unie: cmap_sec->val+expc->type */
  sprintf (*seqtemp, "%-*.*s", lineleng, lineleng, expc->seq+abscol);

  /* ecriture de la ligne */
  WRITE_Line (disp->pan_ali
              , *(disp->col2pix+disp->curcolbeg)
              , *(disp->row2pix+disp->currow)
              , *seqtemp, *(disp->cmap_sec->val+expc->type));

  /* nettoyage */
  memset (*seqtemp, ' ', lineleng+1);

  return;
}
```

Annexe E La bibliothèque « biolcp »

E1 Blastio.h

```
#ifndef BLASTIO_H
#define BLASTIO_H

/*** CONSTANTES SYMBOLIQUES ***/
#define csBlastFline 200+1 /* long d'une ligne d'un fichier BLAST, +1 pour le '\n' */

/*
 * Implemented BLAST tools
 *
 * BLASTIO_IsItABlastResultFile : return 1 if the file is a BLAST query result file
 * BLASTIO_FresultReadLibraryName : read the name of the library in a BLAST result file
 * BLASTIO_FresultGoToSeqNamesBlock : put the BLAST result stream at the beginning of the
protein names block
 */
extern int BLASTIO_IsItABlastResultFile (char* fn_res);
extern void BLASTIO_FresultReadLibraryName (FILE* fp_res, char* libname, long maxlen);
extern int BLASTIO_FresultGoToSeqNamesBlock (FILE* fp_res);

#endif /* BLASTIO_H */
```

E2 Dsspio.h

```
#ifndef DSSPIO_H
#define DSSPIO_H

/*** CONSTANTES SYMBOLIQUES ***/
#define csDsspFline 200+1 /* long d'une ligne d'un fichier DSSP, +1 pour le '\n' */
#define csDsspIDlen 20 /* long d'un seq id d'un fichier DSSP */
#define csDsspHEADlen 100 /* long d'un seq header d'un fichier DSSP */

#define csDsspFileHeaderId "Secondary Structure Definition by the program DSSP"
#define csDsspFileHeaderId_DSSPftp "SECONDARY STRUCTURE DEFINITION BY THE PROGRAM DSSP"
#define csDsspDataBlockId "RESIDUE AA STRUCTURE"

#define errDsspStd           1
#define errDsspOpenFile              2
#define errDsspFileDataLack  3
#define errDsspNotEnoughMem  4
#define errDsspIncreaseChainNb       5
#define errDsspAllocChain    6
#define errDsspIncreaseChainLen      7

/*** TYPEDEF ***/
typedef struct lcpdsspiochain* Lcp_DsspIOChainPtr;
typedef struct lcpdsspiochain {
       char* id;
       char* comments;
       char* prim;
       char* sec;

       long len;
       long lenmax;

       int (*increaselen) (Lcp_DsspIOChainPtr, long);
       void (*free) (void*);
       } Lcp_DsspIOChain;

typedef struct lcpdsspiodata {
       Lcp_DsspIOChain* chains;
       int chainb;
       int chainbmax;

       void (*free) (void*);
       } Lcp_DsspIOData;

/*
 * Implemented DSSP tools
 *
 * DSSPIO_AllocMainData : allocating main DSSP IO data with 'chainb' of length 'chainlen',
return NULL if error
 * DSSPIO_IsItADsspStream : return 1 if the stream is a DSSP stream, return >0 if error
```

```
 * DSSPIO_IsItADsspFile : return 1 if the file is a DSSP secondary file, return >0 if error
 * DSSPIO_ReadDataFromStream : reading a DSSP secondary stream, be care it is a DSSP stream,
return NULL if error
 * DSSPIO_ReadDataFromFile : reading a DSSP secondary file, be care it is a DSSP file, return
NULL if error
 */
Lcp_DsspIOData* DSSPIO_AllocMainData (int chainb, long chainlen);
int DSSPIO_IsItADsspStream (FILE* fp_dssp);
int DSSPIO_IsItADsspFile (char* fn_dssp);
Lcp_DsspIOData* DSSPIO_ReadDataFromStream (FILE* fp_dssp, Lcp_DsspIOData* dsspio);
Lcp_DsspIOData* DSSPIO_ReadDataFromFile (char* fn_dssp, Lcp_DsspIOData* dsspio);

#endif /* DSSPIO_H */
```

E3 Fastaio.h

```
#ifndef FASTAIO_H
#define FASTAIO_H

/*** CONSTANTES SYMBOLIQUES ***/
#define ProtIO_FastaSeqCharAllow "abcdefghiklmnpqrstvwxyzABCDEFGHIKLMNPQRSTVWXYZ"
#define ProtIO_FastaDbFormat 1000 /* big to avoid similarity with readseq or seqio */
#define csFastaFline 80+1 /* long d'une ligne d'un fichier FastA, +1 pour le '\n' */
#define csFastaIDlen 20 /* long d'une seq id d'un fichier FastA, +1 pour le '\n' */

#define csFastaResultFileSeqBlockId "The best score"

/*** TYPEDEF ***/
typedef struct lcpfastaioseq {
       char* id;
       char* comments;
       char* bankname;
       char* seq;

       void (*free) (void*);
       void (*free_elt) (void*);
       } Lcp_FastaIOSeq;

/*
 * Implemented fasta tools
 *
 * FASTAIO_AllocFastaIOSeq : return a Lcp_FastaIOSeq struct type, or NULL if error
 * FASTAIO_IsItACompactFastaDB : return 1 if the file is a compacted Pearson/Fasta format
sequence databank
 * FASTAIO_IsItAFastaResultFile : return 1 if the file is a Fasta query result file
 * FASTAIO_GoToNextEntry : go to the next databank entry
 * FASTAIO_FastReadSeq : Read a seq paragraph (comments + seq). Doesn't test any memory size.
Return sequence length
 * FASTAIO_ReadProtId : read sequence id and skip comments
 * FASTAIO_ReadProtIdAndComments : read sequence id and comments
 * FASTAIO_GetSeq : get the sequence lines in a dynamic structure
 * FASTAIO_GetSeqFromId : extracts a sequence using its identifier
 * FASTAIO_FresultReadLibraryName : read the name of the library in a fasta result file
 * FASTAIO_FresultGoToSeqNamesBlock : put the fasta result file at the beginning of the
protein names block
 */
Lcp_FastaIOSeq* FASTAIO_AllocFastaIOSeq (void);
int FASTAIO_IsItACompactFastaDB (char* fn_db);
int FASTAIO_IsItAFastaResultFile (char* fn_fasres);
int FASTAIO_GoToNextEntry (FILE* fp_in);
long FASTAIO_FastReadSeq (FILE* fp_in, char* comments, char* seq);
int FASTAIO_ReadProtId (FILE* fp_db, char* protid);
int FASTAIO_ReadProtIdAndComments (FILE* fp_in, char* protid, char** comments);
char* FASTAIO_GetSeq (FILE* fp_in);
Lcp_FastaIOSeq* FASTAIO_GetSeqFromId (FILE* fp_in, Lcp_FastaIOSeq* curseq, char* id);
void FASTAIO_FresultReadLibraryName (FILE* fp_fasres, char* libname, long maxlen);
int FASTAIO_FresultGoToSeqNamesBlock (FILE* fp_fasres);

#endif /* FASTAIO_H */
```

E4 Lcphelp.h

```
#ifndef LCP_HELP_H
#define LCP_HELP_H

/*** INCLUDE ***/
```

```
/*** SYMBOLIC CONSTANT ***/
#define csHelpFlineLen 200
#define csHelpFileId "help file"

#define csHelpNoError 0
#define csHelpErrNonExistFile 1
#define csHelpErrCantOpenFile       2
#define csHelpErrNonHelpFile  3

/*** VARIABLES GLOBALES ***/

/* FUNCTION PROTOTYPE
*
* HELP_OpenHelpFile: open help file, if error return NULL
* HELP_IsHelpFile: test if fn_help is a help filename, return error code csHelpErrXxx
* HELP_IsChapterTitleLine: test if 'fline' is a chapter title line (return 1), return 0 else
* HELP_ScanChapterTitle: extract chapter title from the string 'chapid', a help chapter
identifier
* HELP_GoToChapter: go to help chapter number 'chapnum', included help identifier line; if
error return 1
* HELP_ReadChapterTitle: read help chapter title in help file 'fn_help'
*/

FILE* HELP_OpenHelpFile (char* fn_help);
int HELP_IsHelpFile (char* fn_help);
int HELP_IsChapterTitleLine (char* fline);
int HELP_IsParagraphTitleLine (char* fline);
int HELP_IsFormattedTextLine (char* fline);
int HELP_IsTaggedLine (char* fline);
char* HELP_ScanChapterTitle (char* chapid);
char** HELP_ReadChapterTitle (char* fn_help);
int HELP_GoToChapter (FILE* fp_help, int chapnum);

#endif /* LCP_HELP_H */
```

E5 Lcpio.h

```
#ifndef IBCP_IO_H
#define IBCP_IO_H

#include <stdio.h>

/*** SYMBOLIC CONSTANTS ***/
#define csFmanErrBadRights 1
#define csFmanErrOlderFile 2

/*** MPSA FILE MANAGING CODE ***/
#define mpsaFileBanner       "MPSA code"
#define mpsaFileBannerEnd    ':'                                        /* must be a
caracter */

#define mpsaFileLineLen 200
#define mpsaFileSecBlockLen 60

#define mpsaFileAli                1
#define mpsaFileAliStr       "alignment"
#define mpsaFileSec                2
#define mpsaFileSecStr       "secondary"
#define mpsaFilePhysche                    3
#define mpsaFilePhyscheStr   "physchem"
#define mpsaFileProscan                    4
#define mpsaFileProscanStr   "proscan"
#define mpsaFilePattin             5
#define mpsaFilePattinStr    "pattinprot"

/* secondary files */
#define mpsaFileSecTypeUnk          -1
#define mpsaFileSecTypeUnkStr       "unknown"
#define mpsaFileSecTypeScore        1                          /* vertical mode: sec
scores... prim */
#define mpsaFileSecTypeScoreStr             "score"
#define mpsaFileSecTypeSeq                  2                         /* horizontal mode:
prim sec prim sec .... */
#define mpsaFileSecTypeSeqStr       "sequence"
```

```
/* FUNCTION PROTOTYPES
*
* FMAN_IsBinaryAvailable: return 1 if true, 0 else
* FMAN_ExistingFile: return 1 if true, 0 else
* FMAN_ExistingDir: return 1 if true, 0 else
* FMAN_WritableFile: return 1 if true, 0 else
* FMAN_WritableDir: return 1 if true, 0 else
* FMAN_FilenameFromPath: extract filename from a pathway
* FMAN_DirnameFromPath:extract dirname from a pathway
* FMAN_CopyFile: copy file fn_old in new file called fn_new
*
* MPSA file format managing
* FMAN_MpsaFileGetBannerLine: return NULL if MPSA file banner line not found, else return this
line, fline has linelen-1 max char
* FMAN_MpsaFileSec: return secondary identifier array or NULL
* FMAN_MpsaFileAli: return 1 if fp_cur isn't a MPSA sec stream
*/

int     FMAN_IsBinaryAvailable (char* binname);
int     FMAN_ExistingFile (char* filename);
int     FMAN_ExistingDir (char* dirname);
int     FMAN_WritableFile (char* fname); /* return csFmanErrxxxx if error, 0 else */
int     FMAN_WritableDir (char* dname); /* return csFmanErrxxxx if error, 0 else */
char* FMAN_FilenameFromPath (char* path, char dirseparator);
char* FMAN_DirnameFromPath (char* path, char dirseparator);
int     FMAN_CopyFile (char* fn_old, char* fn_new);

/** MPSA file format managing **/
char* FMAN_MpsaFileGetBannerLine (FILE* fp_cur, char* fline, int linelen);
char** FMAN_MpsaFileSec (FILE* fp_cur, int* type, int *secnb);
int FMAN_MpsaFileAli (FILE* fp_cur);

#endif /* IBCP_IO_H */
```

E6 Lcputil.h

```
#ifndef IBCP_UTIL_H
#define IBCP_UTIL_H

/*** SYMBOLIC CONSTANTS ***/

/*** GLOBAL VARIABLES ***/

/* FUNCTION PROTOTYPES
*
* MEM_Free2DArray: free a 2D array, must be finished by NULL
* MEM_AllocCharArray: allocate a char 2D array, must be finished by NULL
* GetMainGraduation: get opt graduation value on this range
* GetSubGraduation:
*
* MATH_Percent: percent of part versus all
* GetNbDig: return number of digit need to print value
*
* GetConsensus : return consensus of list, which size is row x col and gap nul element, get
consensus for each column
*/

void   MEM_Free2DArray (void** arr2free);
char** MEM_AllocCharArray (long nbcol, long nbrow);
double GetMainGraduation (double range);
double GetSubGraduation (double range);

double MATH_Percent (double part, double all);
int    GetNbDig (double value);

int GetConsensus (char* cons, char* percent, char** list, long col, long row, int gap);

#endif /* IBCP_UTIL_H */
```

E7 Proscan.h

```
#ifndef PROSCAN_H
#define PROSCAN_H

/*** PROSCAN ERROR TYPE ***/
#define errSearchCritUnknown               1001 /* unknown search criterion */
```

```
#define errSearchCritSimilBadRange          1002 /* similarity should be [1,100] */
#define errSearchCritMismatchBadRange       1003 /* mismatch should be [1,4] */
#define errResultTooMuchLine                1004 /* mismatch should be [1,4] */

/*** SYMBOLIC CONSTANT ***/
#define csPROSCANvers "3.0"
#define csPROSCANdate "April 8 1998"


#define csProscanMismatchMin 1
#define csProscanMismatchMax 4
#define csProscanSimilMin 1
#define csProscanSimilMax 100
#define csProscanNoDoc "none"

/*** PROTOTYPES ***/
int ProtMotif (char* protein, char* searchcrit, char* pattern);
int Proscan (char* fn_prot, char* searchcrit, char* fn_prositedb, char* fn_prositedoc);
int Pattinprot (char* fn_pattern, char* searchcrit, char* fn_protdb);

extern int PATTINPROT_IsItAPattinprotResultFile (char* fn_res);
extern void PATTINPROT_FresultReadLibraryName (FILE* fp_res, char* libname, long maxlen);
extern int PATTINPROT_FresultGoToNextProt (FILE* fp_pattinprot);
extern int PATTINPROT_FresultGoToFirstProt (FILE* fp_pattinprot);

#endif /* PROSCAN_H */
```

E8 Physchem.h

```
#ifndef IBCP_PHYSCHEM_H
#define IBCP_PHYSCHEM_H

#include <stdio.h>

/* Physico-chemical method name */
#define csHydrophilicityHoppWoodsName       "Hydrophilicity (Hoop & Woods, 1981)"
#define csHydropathyKyteDoolittleName       "Hydropathy (Kyte & Doolittle, 1982)"
#define csFlexibilityKarplusSchulzName      "Flexibility (Karplus & Schulz, 1985)"
#define csAntigenicityParkerName     "Antigenicity (Parker et al., 1986)"
#define csAccessibilityJaninName     "Accessibility (Janin, 1979)"
#define csTransmembArgosName         "Transmembranous helices (Rao & Argos, 1986)"
#define csAntigenicityWellingName    "Antigenicity (Welling, 1985)"

/* Physico-chemical method GLOBAL type */
#define csHydrophilicityHoppWoods    1
#define csHydropathyKyteDoolittle    2
#define csFlexibilityKarplusSchulz   3
#define csAntigenicityParker         4
#define csAccessibilityJanin         5
#define csTransmembArgos                    6
#define csAntigenicityWelling        7

/* Physico-chemical method LOCAL type */
#define csAntigenicityParker_Hplc                                       8
#define csAntigenicityParker_AccessibilityJanin                 9
#define csAntigenicityParker_FlexibilityKarplusSchulz      10

/* Physico-chemical method graph value to data value */
#define csHoppWoodsThresh    0.0
#define csKyteDoolittleThresh 0.0
#define csKarplusSchulzThresh 0.0
#define csParkerThresh       0.0
#define csJaninThresh        0.0
#define csArgosThresh        1.12
#define csWellingThresh      0.0

/* Physico-chemical method graph value to data value */
#define csHoppWoodsDatatoy   100.0
#define csKyteDoolittleDatatoy       100.0
#define csKarplusSchulzDatatoy       100.0
#define csParkerDatatoy              10.0
#define csJaninDatatoy       100.0
#define csArgosDatatoy       100.0
#define csWellingDatatoy     100.0

/*
```

```
 * TYPEDEF
 */

typedef struct ibcpphyschemio {
       char* primname; /* primary name */
       char* primseq; /* primary sequence */
       char* primcomment; /* primary comments */
       long primlen; /* primary length */

       int win;
       int methnb; /* method number */
       int* type; /* cf over */
       char** name; /* cf over */
       double** score; /* calculated scores*/
}Ibcp_PhysChemIO;

/* FUNCTIONS PROTOTYPES
*       PHYSCHEM_WriteMpsaProfileFile: write MPSA format for PC profiles;
*/
int PHYSCHEM_ShiftingSegmentMethod (int methtype, char* primseq, int compwin, double* score);
int PHYSCHEM_KarplusSchulzFlexibility (int methtype, char* primseq, int compwin, double*
score);
int PHYSCHEM_ParkerAntigenicity (int methtype, char* primseq, int compwin, double* score);
int PHYSCHEM_WriteMpsaProfileFile (FILE* fp_pc, Ibcp_PhysChemIO* pcio);

/* Physico-chemical method GLOBAL data */
#define csPhysChemMax 7
#define csPhysChemTypecsHydrophilicityHoppWoods, csHydropathyKyteDoolittle \
                                          , csFlexibilityKarplusSchulz, csAntigenicityParker
\
                                          , csAccessibilityJanin, csTransmembArgos \
                                          , csAntigenicityWelling
#define csPhysChemNamecsHydrophilicityHoppWoodsName, csHydropathyKyteDoolittleName \
                                          , csFlexibilityKarplusSchulzName,
csAntigenicityParkerName \
                                          , csAccessibilityJaninName, csTransmembArgosName \
                                          , csAntigenicityWellingName
#define csPhysChemThresh     csHoppWoodsThresh, csKyteDoolittleThresh \
                                          , csKarplusSchulzThresh, csParkerThresh \
                                          , csJaninThresh, csArgosThresh \
                                          , csWellingThresh
#define csPhysChemDatatoy    csHoppWoodsDatatoy, csKyteDoolittleDatatoy \
                                          , csKarplusSchulzDatatoy, csParkerDatatoy \
                                          , csJaninDatatoy, csArgosDatatoy \
                                          , csWellingDatatoy
#define csPhysChemProcPHYSCHEM_ShiftingSegmentMethod, PHYSCHEM_ShiftingSegmentMethod \
                                          , PHYSCHEM_KarplusSchulzFlexibility,
PHYSCHEM_ParkerAntigenicity \
                                          , PHYSCHEM_ShiftingSegmentMethod,
PHYSCHEM_ShiftingSegmentMethod \
                                          , PHYSCHEM_ShiftingSegmentMethod


#endif /* IBCP_PHYSCHEM_H */
```

E9 Secondary.h

```
#ifndef IBCP_SECONDARY_H
#define IBCP_SECONDARY_H

#include <stdio.h>

/* sec statee */
#define csSecStateMax 8+1 /* nb max d'etats de struct sec: 8 + '?' pour l'indecision */
#define csSecStateOrder "HETCSBGI?" /* ordre des struct sec */
#define csSecStateName"Alpha Helix", "Beta Sheet", "Turn" \
                                          , "Coil", "Bend", "Beta Bridge" \
                                          , "3-10 Helix", "Pi Helix", "doubtful"

#define csSecStateDoubtful '?' /* caract utilise en cas d'indecision */
#define csSecStateGap ' ' /* caract utilise pour les gaps */

/* sec methods: ! order is important, put a new one at the end */
#define csSecUNK      0
#define csSecGOR      1
#define csSecGIB      2
```

```
#define csSecLEV      3
#define csSecDPM      4
#define csSecSOPM     5
#define csSecSOPMA    6
#define csSecCFA      7
#define csSecPHD      8
#define csSecCONS     9
#define csSecXRAY     10
#define csSecGOR4     11
#define csSecPREDATOR 12
#define csSecSIMPA    13
#define csSecDSSP     14
#define csSecALL      15
#define csSecMAX      15

#define csSecUNKname  "Unknown sec."
#define csSecGORname  "GOR 1"
#define csSecGIBname  "GOR 2"
#define csSecLEVname  "Homologue"
#define csSecDPMname  "Double Prediction"
#define csSecSOPMname "SOPM"
#define csSecSOPMAname"SOPMA"
#define csSecCFAname  "Chou-Fasman"
#define csSecPHDname  "PHD"
#define csSecCONSname "Sec. Cons."
#define csSecXRAYname "Other sec."
#define csSecGOR4name "GOR 4"
#define csSecPREDATORname    "Predator"
#define csSecSIMPAname"SIMPA 96"
#define csSecDSSPname "PDB structure"

/* be carefull: only one word for fsuff */
#define csSecUNKfsuff        "UNKSEC"
#define csSecGORfsuff        "GOR1"
#define csSecGIBfsuff        "GOR2"
#define csSecLEVfsuff        "LEV"
#define csSecDPMfsuff        "DPM"
#define csSecSOPMfsuff       "SOPM"
#define csSecSOPMAfsuff              "SOPMA"
#define csSecCFAfsuff        "CFA"
#define csSecPHDfsuff        "PHD"
#define csSecCONSfsuff       "CONS"
#define csSecXRAYfsuff       "XRAY"
#define csSecGOR4fsuff       "GOR4"
#define csSecPREDATORfsuff   "PREDA"
#define csSecSIMPAfsuff              "SIMPA"
#define csSecDSSPfsuff       "DSSP"

#define csSecUNKstord csSecStateOrder
#define csSecGORstord "HETC?"
#define csSecGIBstord "HEC?"
#define csSecLEVstord csSecStateOrder
#define csSecDPMstord "HETC?"
#define csSecSOPMstord"HETC?"
#define csSecSOPMAstord      "HETC?"
#define csSecCFAstord "HETC?"
#define csSecPHDstord "HEC?"
#define csSecCONSstordcsSecStateOrder
#define csSecXRAYstordcsSecStateOrder
#define csSecGOR4stord"HEC?"
#define csSecPREDATORstord   "HEC?"
#define csSecSIMPAstord      "HEC?"
#define csSecDSSPstordcsSecStateOrder

/* BE CAREFULL: all these 4 define must have the same meth order */
#define csSecAllType csSecUNK,  csSecGOR, csSecGIB, csSecLEV \
                                    , csSecDPM , csSecSOPM, csSecSOPMA, csSecCFA \
                                    , csSecPHD, csSecCONS, csSecXRAY \
                                    , csSecGOR4, csSecPREDATOR, csSecSIMPA \
                                    , csSecDSSP
#define csSecAllName csSecUNKname,  csSecGORname, csSecGIBname, csSecLEVname \
                                    , csSecDPMname, csSecSOPMname, csSecSOPMAname,
csSecCFAname \
                                    , csSecPHDname, csSecCONSname, csSecXRAYname \
                                    , csSecGOR4name, csSecPREDATORname, csSecSIMPAname \
                                    , csSecDSSPname
#define csSecAllFsuff csSecUNKfsuff,  csSecGORfsuff, csSecGIBfsuff, csSecLEVfsuff \
```

```
                                    , csSecDPMfsuff, csSecSOPMfsuff, csSecSOPMAfsuff,
csSecCFAfsuff \
                                    , csSecPHDfsuff, csSecCONSfsuff, csSecXRAYfsuff \
                                    , csSecGOR4fsuff, csSecPREDATORfsuff, csSecSIMPAfsuff \
                                    , csSecDSSPfsuff
#define csSecAllStatOrder csSecUNKstord,  csSecGORstord, csSecGIBstord, csSecLEVstord \
                                    , csSecDPMstord, csSecSOPMstord, csSecSOPMAstord,
csSecCFAstord \
                                    , csSecPHDstord, csSecCONSstord, csSecXRAYstord \
                                    , csSecGOR4stord, csSecPREDATORstord, csSecSIMPAstord \
                                    , csSecDSSPstord

/*** TYPEDEF ***/
typedef struct lcpsecondary {
       char* prim;
       char* sec;
       long len;

       int datatype; /* score or seq */
       int methtype;
       char* primname;
       char* primcomments;
       char* secname;

       int stnb;
       char storder[csSecStateMax+1];
       int** score;

       char* predparam; /* parameters used during the prediction iteration and wrote below
scor min/max */
       } Lcp_Secondary;


/* FUNCTION PROTOTYPES
*
* SEC_FreeSecData: free only pointer field of current Lcp_Secondary
* SEC_MpsaGetSizeFromFline: extract sec len, sec state nb, sec state order
* SEC_Read: secondary structure generic reading function, return NULL in case of error
* SEC_ReadMpsaScoreFile: read prim & sec, they need to be alloc with seclen+1 char
* SEC_ReadMpsa: reading MPSA file containing one(n) secondary(ies)
* SEC_ReadPredator: reading Predator sec. pred. file format
* SEC_ReadSIMPA96: reading SIMPA 96 sec. pred. file format
* SEC_WriteMpsaConsensusFile: writing file containing one(n) secondary(ies)
*/
void SEC_FreeSecData (Lcp_Secondary* cursec);
int SEC_MpsaGetSizeFromFline (char* fline, Lcp_Secondary* cursec);
Lcp_Secondary* SEC_Read (FILE* fp_sec, Lcp_Secondary* cursec, int* secnb);
long SEC_ReadMpsaScoreFile (FILE* fp_cur, Lcp_Secondary* cursec, int secnb);
Lcp_Secondary* SEC_ReadMpsa (FILE* fp_sec, Lcp_Secondary* cursec, int* secnb);
int SEC_ReadPredator (FILE* fp_sec, Lcp_Secondary* cursec);
int SEC_ReadSIMPA96 (FILE* fp_sec, Lcp_Secondary* cursec);
void SEC_WriteMpsaConsensusFile (FILE* fp_cons, Lcp_Secondary* seclist, Lcp_Secondary* secons,
long seclen, int secnb);

#endif /* IBCP_SECONDARY_H */
```

E10 Secpred.h

```
#ifndef IBCP_SECPRED_H
#define IBCP_SECPRED_H

/*** GARNIER
*              Garnier, J., Osguthorpe, D.J. & Robson, B. (1978) Analysis of
*              the accuracy and implication of simple methods for predicting
*              the secondary structure of globular proteins. J. Mol. Biol.,
*              120, 97-120
*/
#define csGorStateNb   4 /* nb d'etats de struct sec predits */
#define csGorStatePriority    "CHET" /* ordre de priorite decroissante des etats de struct sec
predits */
#define csGorStateWriteOrder "HETC" /* ordre d'impression des etats de struct sec predits */
int Garnier (char* fn_in, char* fn_out);

/*** GIBRAT
*              Gibrat, J.F., Garnier, J. & Robson, B. (1987) Further
*              developments of protein secondary structure prediction using
```

```
*              information theory. New parameters and consideration of residue
*              pairs. J. Mol. Biol., 198, 425-443.
*/
#define csGibStateNb   3 /* nb d'etats de struct sec predits */
#define csGibStatePriority    "CHE" /* ordre de priorite decroissante des etats de struct sec
predits */
#define csGibStateWriteOrder "HEC" /* ordre d'impression des etats de struct sec predits */
int Gibrat (char* fn_in, char* fn_out);

/*** LEVIN
*              Levin, J.M., Robson, B. & Garnier, J. (1986) An algorithm for
*              secondary structure determination in protein based on sequence
*              similarity. FEBS Lett., 205, 303-308.
*/
#define csLevStateNb   8 /* nb d'etats de struct sec predits */
#define csLevStatePriority    "CHETSBGI" /* ordre de priorite decroissante des etats de struct
sec predits */
#define csLevStateWriteOrder "HETCSBGI" /* ordre d'impression des etats de struct sec predits
*/
int Levin (char* fn_in, char* fn_out);


#endif /* IBCP_SECPRED_H */
```

E11 mpsaweb.h

```
#ifndef MPSA_WEB_H
#define MPSA_WEB_H

#define csWebMpsa "bmw.ibcp.fr"

#define  csCgibinClustalw   "mpsa_clustalw.pl"
#define  csCgibinMultalin   "mpsa_multalin.pl"
#define  csCgibinDatabank   "mpsa_db.pl"
#define  csCgibinProsite   "mpsa_prosite.pl"
#define  csCgibinPattinprot   "mpsa_pattinprot.pl"
#define  csCgibinFasta   "mpsa_fasta.pl"
#define  csCgibinBlast   "mpsa_blast.pl"
#define  csCgibinGetSecPredMeth   "mpsa_secmethavail.pl"
#define  csCgibinSecPred   "mpsa_secpred.pl"
#define  csCgibinMpsaLastHelp   "mpsa_help.pl"

#define csCgiTagDbAvail "avail"
#define csCgiTagDbDownload "download"
#define csCgiTagDbSubset "subset"

#define csCgiServiceDb "databank"
#define csCgiServiceAli "alignment"
#define csCgiServiceGoodResponse "OK"


/*
 * protein sequence analysis methods
 */
extern int WWW_SubmitClustal (char* webserver, char* fn_bseq, char* fn_ali, char* arg);
extern int WWW_SubmitMultalin (char* webserver, char* fn_bseq, char* fn_ali, char* arg);

extern int WWW_GetRemoteBankList (char* webserver, FILE* fp_tmp);
extern int WWW_GetRemoteBankSubset (char* webserver, FILE* fp_tmp
                                                                , char *bankname, char
*seqid);

extern int WWW_SubmitProscan (char* webserver, char* fn_out
                                    , char* seqid, char* seq, char *comment
                                    , int simil, int mismatch
                                    , int flg_doc);
extern int WWW_SubmitPattinprot (char* webserver, char* fn_out
                                    , char* pattern, char* pattname, char* comment
                                    , int simil, int mismatch
                                    , char* bank);
extern int WWW_SubmitFasta (char* webserver, char* fn_out
                                    , char* seqid, char* seq, char *comment, char* bank);
extern int WWW_SubmitBlast (char* webserver, char* fn_out
                                    , char* seqid, char* seq, char *comment, char* bank);

extern int WWW_GetRemoteSecPredMethList (char* webserver, FILE* fp_tmp);
```

```
extern int WWW_SubmitSecPred (char* webserver, char* fn_out
                                    , char* seqid, char* seq, char *comment
                                    , char* meth);

#endif /* MPSA_WEB_H */
```

Annexe F Echelles des profils physico-chimiques utilisés.

	Hydrophilie de Hopps et Woods (P 106)	Rao et Argos (P 158)	Flexibilité de Karplus et Schulz (P 116)			Antigénicité de Welling *et al.* (P 185)	Hydrophobie de Kyte et Doolittle (P 120)	Accessibilité au solvant de Janin (P 110)
			0 voisin rigide	1 voisin rigide	2 voisins rigides			
A	-0.5	1.36	1.041	0.946	0.892	0.115	1.8	0.6
C	-1.0	1.27	0.960	0.878	0.925	-0.120	2.5	0.2
D	3.0	0.11	1.033	1.089	0.932	0.065	-3.5	2.7
E	3.0	0.25	1.094	1.036	0.933	-0.071	-3.5	3.2
F	-2.5	1.57	0.930	0.912	0.914	-0.141	2.8	0.5
G	0.0	1.09	1.142	1.042	0.923	-0.184	-0.4	0.6
H	-0.5	0.68	0.982	0.952	0.894	0.312	-3.2	1.2
I	-1.8	1.44	1.002	0.892	0.872	-0.292	4.5	0.3
K	3.0	0.09	1.093	1.082	1.057	0.206	-3.9	20.6
L	-1.8	1.47	0.967	0.961	0.921	0.075	3.8	0.4
M	-1.3	1.42	0.947	0.862	0.804	-0.385	1.9	0.5
N	0.2	0.33	1.117	1.006	0.930	-0.077	-3.5	2.3
P	0.0	0.54	1.055	1.085	0.932	-0.053	-1.6	1.8
Q	0.2	0.33	1.165	1.028	0.885	-0.011	-3.5	3.2
R	3.0	0.15	1.038	1.028	0.901	0.058	-4.5	9.0
S	0.3	0.97	1.169	1.048	0.923	-0.026	-0.8	1.2
T	-0.4	1.08	1.073	1.051	0.934	-0.045	-0.7	1.4
V	-1.5	1.37	0.982	0.927	0.913	-0.013	4.2	0.3
W	-3.4	1.0	0.925	0.917	0.803	-0.114	-0.9	0.6
Y	-2.3	0.83	0.961	0.930	0.837	0.013	-1.3	2.0

Les acides aminés rigides d'après Karplus et Schulz sont les suivant : ALHVYIFCWM.

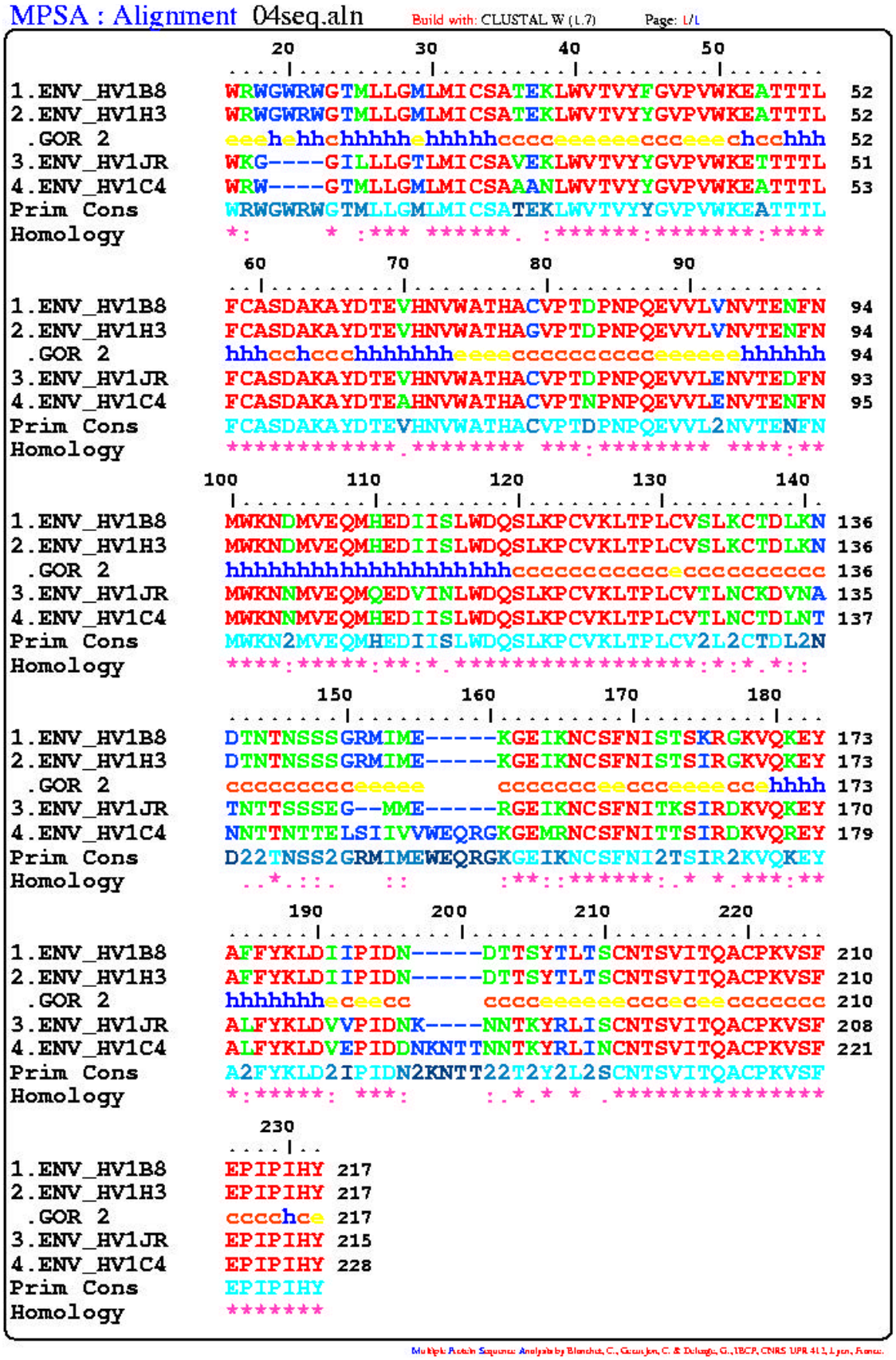
	Hydrophilie de Parker *et al.* (P 150)				
	HPLC	Accessibilité de Janin	Flexibilité de Karplus et Schulz		
			0 voisin rigide	1 voisin rigide	2 voisins rigides
A	2.1	+2.7	-0.5	-2.6	-3.0
C	1.4	-10.0	-7.1	-8.6	-0.4
D	10.0	8.4	-1.1	10.0	0.1
E	7.8	8.9	3.8	5.3	0.7
F	-9.2	+0.5	-9.6	-5.6	-1.2
G	5.7	2.3	7.8	5.8	-0.5
H	2.1	6.7	-5.3	-2.1	-2.8
I	-8.0	-3.4	-3.7	-7.3	-4.6
K	5.7	10.0	3.8	9.4	10.0

	Hydrophilie de Parker *et al.* (P 150)				
	HPLC	**Accessibilité de Janin**	**Flexibilité de Karplus et Schulz**		
			0 voisin rigide	**1 voisin rigide**	**2 voisins rigides**
L	-9.2	-0.3	-6.5	-1.3	-0.7
M	-4.2	+1.9	-8.2	-10.0	-9.9
N	7.0	8.4	5.7	2.7	0.0
P	2.1	+7.5	0.6	9.6	0.1
Q	6.0	8.7	9.6	4.6	-3.5
R	4.2	9.8	-0.7	4.6	-2.3
S	6.5	6.7	10.0	6.4	-0.5
T	5.2	7.1	2.1	6.6	0.3
V	-3.7	-2.5	-5.3	-4.2	-1.3
W	-10.0	+3.2	-10.0	-5.1	-10.0
Y	-1.9	+8.0	-7.0	-4.0	-7.3

Parker *et al.* définissent une échelle d'hydrophilie basée sur des coefficients de rétention de peptides modèles sur HPLC et utilisent les échelles de Janin (P 110) et de Karplus & Schulz (P 116). Ils norment ces échelles entre –10 et +10.

Annexe G MPSA, les sauvegardes mises en forme

G1 Impression PostScript d'un alignement



G2 Un alignement au format RTF

```
Generated with MPSA 0.8b
Multiple Protein Sequence Analysis
Blanchet, C., Geourjon, C. & Deleage, G.
Pôle Bioinformatique Lyonnais
IBCP (CNRS-UPR 412)
Lyon, FRANCE

Contact: C. Blanchet, c.blanchet@ibcp.fr, http://www.ibcp.fr/~blanchet, fax: +33 (0)4 72 76 90
50

Multiple alignment engine: CLUSTAL W version 1.7

Alignment parameters:
ENV_HV1B8 (# 1) len=851 aa
ENV_HV1H3 (# 2) len=856 aa
Sec: GOR 2 (len=856) Sec. state rate ( H=43.2%% E=24.9%% C=31.9%% ?=0.0%%) Parameters (Build-
in). Relative primary: ENV_HV1H3 (# 2)
ENV_HV1JR (# 3) len=848 aa
ENV_HV1C4 (# 4) len=868 aa

Colour-codes used to display the Alignment :
Primary :
       this color indicates positions which have a single, fully conserved residue
       this color indicates that one of the following 'strong' groups (STA, NEQK, NHQK, NDEQ,
QHRK, MILV, MILF, HY, FYW) or 'weaker' groups (CSA, ATV, SAG, STNK, STPA, SGND, SNDEQK,
NDEQHK, NEQHRK, FVLIM, HFY) is fully conserved.
       this color indicates positions which have no particularity
Secondary :
       Alpha Helix (h or H)
       Beta Sheet (e or E)
       Turn (t or T)
       Coil (c or C)
       Bend (s or S)
       Beta Bridge (b or B)
       3-10 Helix (g or G)
       Pi Helix (i or I)
       doubtful (? or ?)

                20        30        40        50        60
               ....|.....|.....|.....|.....|.....|.....
1.ENV_HV1B8    WRWGWRWGTMLLGMLMICSATEKLWVTVYFGVPVWKEATTTLFCASDAKA  60
2.ENV_HV1H3    WRWGWRWGTMLLGMLMICSATEKLWVTVYYGVPVWKEATTTLFCASDAKA  60
 .GOR 2        eeehehhchhhhhehhhhhcccceeeeeeccceeechcchhhhhhcchcc  60
3.ENV_HV1JR    WKG----GILLLGTLMICSAVEKLWVTVYYGVPVWKETTTTLFCASDAKA  59
4.ENV_HV1C4    WRW----GTMLLGMLMICSAAANLWVTVYYGVPVWKEATTTLFCASDAKA  61
Primary cons.  WRWGWRWGTMLLGMLMICSATEKLWVTVYYGVPVWKEATTTLFCASDAKA
Homology       *:     * :*** ******. :******:*******:************

                70        80        90       100       110
               ....|.....|.....|.....|.....|.....|.....|.....
1.ENV_HV1B8    YDTEVHNVWATHACVPTDPNPQEVVLVNVTENFNMWKNDMVEQMHEDIIS 110
2.ENV_HV1H3    YDTEVHNVWATHAGVPTDPNPQEVVLVNVTENFNMWKNDMVEQMHEDIIS 110
 .GOR 2        chhhhhhheeeeccccccccccceeeeeehhhhhhhhhhhhhhhhhhhhhh 110
3.ENV_HV1JR    YDTEVHNVWATHACVPTDPNPQEVVLENVTEDFNMWKNNMVEQMQEDVIN 109
4.ENV_HV1C4    YDTEAHNVWATHACVPTNPNPQEVVLENVTENFNMWKNNMVEQMHEDIIS 111
Primary cons.  YDTEVHNVWATHACVPTDPNPQEVVL2NVTENFNMWKN2MVEQMHEDIIS
Homology       ****.******** ***:******** ****:******:*****:**:*.

                120       130       140       150       160
               ....|.....|.....|.....|.....|.....|.....|.....
1.ENV_HV1B8    LWDQSLKPCVKLTPLCVSLKCTDLKNDTNTNSSSGRMIME-----KGEIK 155
2.ENV_HV1H3    LWDQSLKPCVKLTPLCVSLKCTDLKNDTNTNSSSGRMIME-----KGEIK 155
 .GOR 2        hhhhccccccccccceccccccccccccccccceeeee     ccccc 155
3.ENV_HV1JR    LWDQSLKPCVKLTPLCVTLNCKDVNATNTTSSSEG--MME-----RGEIK 152
4.ENV_HV1C4    LWDQSLKPCVKLTPLCVTLNCTDLNTNNTTNTTELSIIVVWEQRGKGEMR 161
Primary cons.  LWDQSLKPCVKLTPLCV2L2CTDL2ND22TNSS2GRMIMEWEQRGKGEIK
Homology       *****************:*:*.*::  ..*.::.   ::      :**::

                170       180       190       200       210
               ....|.....|.....|.....|.....|.....|.....|.....
1.ENV_HV1B8    NCSFNISTSKRGKVQKEYAFFYKLDIIPIDN-----DTTSYTLTSCNTSV 200
2.ENV_HV1H3    NCSFNISTSIRGKVQKEYAFFYKLDIIPIDN-----DTTSYTLTSCNTSV 200
 .GOR 2        cceecccceeeeccehhhhhhhhhhhecceecc     ccccceeeeeecccc 200
```

```
3.ENV_HV1JR     NCSFNITKSIRDKVQKEYALFYKLDVVPIDNK----NNTKYRLISCNTSV 198
4.ENV_HV1C4     NCSFNITTSIRDKVQREYALFYKLDVEPIDDNKNTTNNTKYRLINCNTSV 211
Primary cons.   NCSFNI2TSIR2KVQKEYA2FYKLD2IPIDN2KNTT22T2Y2L2SCNTSV
Homology        ******:.* *.***:***:*****: ***:     :.*.* * .*****

                 220       230
                 ....|.........|..
1.ENV_HV1B8     ITQACPKVSFEPIPIHY 217
2.ENV_HV1H3     ITQACPKVSFEPIPIHY 217
 .GOR 2         ceecccccccccchce 217
3.ENV_HV1JR     ITQACPKVSFEPIPIHY 215
4.ENV_HV1C4     ITQACPKVSFEPIPIHY 228
Primary cons.   ITQACPKVSFEPIPIHY
Homology        ****************
```

Depuis quelques années les projets génomes se multiplient et leur durée de réalisation diminue. Les génomes connus sont de plus en plus nombreux et permettent déjà de mettre en évidence des gènes homologues de différentes espèces. Il est alors possible de supposer la fonction d'un gène par homologie avec ceux d'autres espèces. Car c'est là qu'est le véritable enjeu de tous ces programmes de grand séquençage : identifier les fonctions des protéines codées par ces gènes et leurs implications dans les chemins métaboliques. L'analyse de séquence de protéine prend alors tout son sens. En effet, elle essaye d'attribuer aux protéines codées par ces gènes une fonction et des caractéristiques à partir des données déjà connues. Les banques internationales contenant ces données sont de plus en plus grandes et complexes. Les biologistes ont besoin d'outils efficaces pour travailler sur de très grandes familles de protéines et pour intégrer des données complexes et corrélées.

Dans ce contexte, nous avons développé MPSA (Multiple Protein Sequence Analysis) et NPS@ (Network Protein Sequence Analysis) : un ensemble d'outils et de ressources bioinformatiques dédiées à l'analyse de séquences de protéine. Ces logiciels intègrent plusieurs méthodes nécessaires à l'analyse de séquences de protéine comme la recherche de similarités dans les banques, les alignements multiples, les prédictions de structures secondaires, les prédictions de caractéristiques physico-chimiques. MPSA intègre notamment une méthode, que nous avons mise au point, qui permet de faire des recherches de motifs protéiques dans les banques. Cette méthode améliore le rapport signal/bruit lors d'une recherche en privilégiant l'information biologique des motifs. Les résultats de ces diverses méthodes sont présentés graphiquement et reliés les uns aux autres afin d'accélérer et d'améliorer les processus d'analyse. Un accent particulier a été mis sur la convivialité, la simplification et l'automatisation des tâches fastidieuses pour l'utilisateur.

La conception de MPSA en langage C et l'utilisation de bibliothèques d'une portabilité élevée l'affranchissent des contraintes liées aux architectures matérielles (distribué pour UNIX, Mac et PC). Dans un souci de réutilisabilité du code, les différentes méthodes ont été rassemblées dans une bibliothèque C nommée "biolcp". Cette bibliothèque portable est disponible pour de nouvelles utilisations par d'autres bioinformaticiens. Elle peut être obtenue, tout comme MPSA, sur le site Web de MPSA (http://www.ibcp.fr/mpsa), qui propose également une aide en ligne et les informations sur le développement de MPSA. L'adjonction d'un module client-serveur HTTP (Web) offre à MPSA les potentialités que sont à même de proposer les grands serveurs biologiques : gestion centralisée des données et de leur mise à jour; grande puissance de calcul. La partie serveur-MPSA, implémentée en Perl 5, est disponible sur le site Web de MPSA et peut alors être rapatriée et installée sur un serveur local ou un intranet.

MPSA software and biocomputing web client-server facilities dedicated to protein sequence analysis.

Protein sequence analysis represents the current challenge in retrieving most information contained in data generated by genome projects. It has to face with the handling of databanks with increasing size, and methods with considerable complexity. MPSA (Multiple Protein Sequence Analysis) and NPS@ (Network Protein Sequence Analysis) answer to these constraints with a friendly graphic user interface to several methods from different biological or biophysical fields. In this way, parametring these methods, making them collaborate or displaying their results become very efficient and easy to use for all biologists.

MPSA and NPS@ provide several protein sequence analysis algorithms: (i) biologically significant sites, patterns and profiles scans (ProScan, PattInProt), (ii) homologous protein query (FASTA, BLAST, PSI-BLAST, etc.) in databases (SWISS-PROT, TrEMBL, PIR, non-redundant, PDB, etc.), (iii) multiple alignment (Clustal W, Multalin), (iv) secondary structure prediction (GOR, SOPMA, MLR, PREDATOR, etc.), (v) physico-chemical profile prediction, *etc*. Using MPSA, the biologist does not worry about algorithm collaboration or data formats. Protein sequence analysis methods, included in MPSA, can be processed on a local computer, or on a remote computer through Web connection. With these facilities, the biologist browses databanks daily-updated on remote Web server, and uses up-to-date algorithms delivered by bioinformatic laboratories like as ours for PattInProt or ProScan.

PattInProt and ProScan use an algorithm we develop that allowed the biologist either to scan protein sequence databanks (SWISS-PROT, TrEMBL, *etc.*) with patterns or to scan a sequence for the occurrence of PROSITE patterns. This algorithm emphasize pattern biological information and then increased signal/noise ratio of the scan.

MPSA is written in C_{ANSI} and used libraries with high portability. That is why MPSA is available for Macintosh, PC, UNIX workstations or personal computer with Linux system. In order to provide our methods to the bioinformatic community, we put them into a C library called « biolcp ».

One of the module included in « biolcp » is the MPSAweb client module that provides to MPSA the ability to contact a biological web server through HTTP, like NPS@, which contains the MPSAweb server module. The «biolcp » library (including MPSAweb client) and the MSPAweb server (Perl 5 scripts) can be downloaded as well as MPSA binaries from the MPSA web site at http://www.ibcp.fr/mpsa. This web site also provides biologists with online documentation and release notes. Therefore the biocomputer scientist can download MPSA and the MPSAweb modules in order to install them on his local web server or on an intranet.

Bioinformatique

Bioinformatique - Analyse de séquences - Protéine- Motif protéique – Logiciel - Serveur Web

Laboratoire de Conformation des Protéines
IBCP – CNRS UPR 412. 7, passage du Vercors. 69367 Lyon CEDEX 07