UNIVERSITE GRENOBLE I – JOSEPH FOURIER

UFR D'INFORMATIQUE ET DE MATHEMATIQUES APPLIQUEES

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE I

Discipline : Informatique

présentée et soutenue publiquement

par

Nicolas THIERRY-MIEG

le 17 Octobre 2001

Titre :

Modélisation informatique et analyse prédictive des interactions protéine-protéine chez *Caenorhabditis elegans*

JURY

Rapporteurs :	Richard DURBIN
	François JACQUENET
Examinateurs :	Jacques COHEN
	Paul JACQUET
	Thierry VERNET
	Marc VIDAL
Directeur de thèse :	Laurent TRILLING

Remerciements

Mes travaux de thèse se situent à l'interface entre la biologie et l'informatique. De plus, ils se sont déroulés dans deux localités géographiques : principalement dans l'équipe dirigée par Laurent Trilling au sein du laboratoire Logiciels-Systèmes-Réseaux, à Grenoble, mais aussi à Boston où j'ai été accueilli dans l'équipe de Marc Vidal au cours de trois séjours d'une durée totale de sept mois, initialement au Massachusetts General Hospital Cancer Center puis au Dana Farber Cancer Institute. Je voudrais en premier lieu remercier vivement Laurent Trilling, qui a accepté de diriger cette thèse, s'y est investi sans ménagement au prix d'une reconversion thématique, et a su toujours proposer des pistes prometteuses tout en me laissant totalement libre de choisir ma voie. Je remercie tout aussi chaleureusement Marc Vidal, pour son énergie débordante et sa science exceptionnelle, sans qui le sujet même de cette thèse n'aurait pas eu lieu d'être. Je remercie également Thierry Vernet et Jacques Cohen, qui m'ont apporté un encadrement supplémentaire avec grand enthousiasme, en biologie à Grenoble et en informatique à Boston, respectivement. Je tiens aussi à remercier Paul Jacquet et Ed Harlow, qui m'ont accueilli dans leurs laboratoires. Je remercie tous les membres de mon jury, en particulier Richard Durbin et François Jacquenet qui ont accepté d'être rapporteurs de ma thèse. Je remercie Anne Davy, ainsi que Jean-François Boulicaut et Baptiste Jeudy, avec qui j'ai eu le plaisir de collaborer. Je remercie Proteome Inc., qui m'a donné un accès privilégié aux données d'interaction de la base YPD. Je remercie toutes les personnes que j'ai côtoyé dans les laboratoires à Grenoble et Boston, qui m'ont rendu la vie et le travail plus agréable, en particulier Rose-Ann, Valérie, Pascal et Enzo, ainsi que Michel et Sébastien. Je veux aussi remercier mes parents Danielle et Jean pour leur soutien moral et scientifique. Et bien sûr je remercie mes amis, les montpelliérains et les autres, car pour travailler bien il faut bien se détendre… Merci enfin à Delphine, qui m'a donné Lila, et à Lila, qui m'a appris que le sommeil est un luxe.

TABLE DES MATIERES

1.Introduction

Mon travail de thèse se situe à l'interface entre l'informatique et la biologie. Je suppose que le lecteur est familier avec les notions de base de ces deux disciplines. Ainsi, bien que je m'attache à définir autant que possible les termes techniques avant de les employer, je ne m'étendrai pas dans cet exposé sur le sens des objets biologiques que sont les gènes, l'ADN, l'ARN ou les protéines, tout comme je ne souhaite pas expliciter les notions fondamentales de l'informatique comme l'algorithmique et la complexité, ou bien les systèmes de gestion de bases de données relationnels et objets. Pour une bonne introduction à la biologie, je conseille la lecture d'ouvrages tels que [Wat94].

Dans cette introduction, je présente d'abord brièvement le cadre thématique dans lequel se situe ce travail. J'expose ensuite le sujet de la thèse et l'approche adoptée. Finalement, le plan de ce document est présenté.

1.1 Cadre thématique

Nous vivons, à mon sens, l'une des grandes révolutions scientifiques de l'histoire de l'humanité. L'avènement et le développement des technologies de séquençage de l'ADN à grande échelle, et l'application des techniques de l'ADN recombinant, ont bouleversé notre conception de la biologie et permettent d'envisager une compréhension globale du fonctionnement des organismes vivants. Ainsi, nous disposons des séquences des génomes de la levure *Saccharomyces cerevisiae* [Gof96], du nématode *Caenorhabditis elegans* [Cel98], et plus récemment de la mouche *Drosophila melanogaster* [Ada00] et d'une ébauche du génome humain [Lan01].

Cependant, l'obtention des séquences génomiques brutes n'est pas l'aboutissement des recherches en biologie du génome : bien au contraire, elle en est le point de départ. La seconde étape consiste à identifier les zones codantes, ou gènes. Cette tâche est ardue : la séquence brute s'avère difficile à décoder, surtout chez les eucaryotes où des choix alternatifs de promoteurs, d'épissage et de

polyadénylation, ainsi que la capacité à éditer l'ARN par des modifications post-transcriptionnelles, génèrent de nombreux variants de l'information génomique [Thi01b].

Il s'agit ensuite de déterminer les fonctions des gènes identifiés. Dans cette optique, la génomique fonctionnelle peut fournir très rapidement des indices, par application de techniques expérimentales à grande échelle. Ainsi, depuis la publication de son génome en 1998, plusieurs projets de génomique fonctionnelle concernant *C. elegans* ont vu le jour [Ste01], allant de l'amputation systématique de zones codantes [KOC] ou l'élimination fonctionnelle de gènes par interférence ARN ([Gon00], [Fra00], [Pia00], [Mae01]), jusqu'au développement de micro-arrays globaux ([Rei00], [Hil00], [Jia01]), en passant par la cartographie des interactions protéine-protéine ([Wal00], [Dav01]). Au sein de ces diverses approches, l'informatique joue souvent un rôle fondamental, depuis la génération et le stockage des données expérimentales jusqu'au traitement et à l'analyse des résultats.

Considérons en particulier le problème de la cartographie des interactions protéine-protéine. Rappelons que les protéines sont les principaux effecteurs fonctionnels et structuraux de la cellule, et qu'elles sont susceptibles d'interagir, c'est-à-dire de se lier physiquement. Il apparaît de plus en plus clairement que ces interactions sont extrêmement importantes. En effet, elles jouent un rôle fondamental dans la majorité des processus biologiques, depuis la formation des structures macromoléculaires et des complexes enzymatiques jusqu'à la régulation des voies métaboliques ou la transmission de signaux biologiques. Pour déterminer la fonction d'une protéine, il est donc essentiel de découvrir avec quelles autres protéines celle-ci est susceptible d'interagir. Plusieurs méthodes expérimentales ont été développées afin de tenter de répondre à cette question. Parmi ces méthodes, le système dit "double hybride" [Fie89] s'impose dans les laboratoires comme la technique de choix, en particulier dans le cadre des projets de génomique fonctionnelle. Certaines versions optimisées de ce système s'avèrent en effet applicable à l'échelle d'un génome complet, comme le montrent les études menées sur la levure *Saccharomyces cerevisiae* ([Ito00], [Uet00]).

Il faut cependant relativiser la notion de "haut débit" en génomique fonctionnelle. S'il est bien envisageable d'appliquer une version adaptée du système double hybride à l'échelle d'un génome, la tâche n'en reste pas moins un

travail de longue haleine. Par exemple, le projet de cartographie systématique des interactions chez *C. elegans* ([Wal00], [Dav01]) a été initié en 1998 et est appelé à durer encore plusieurs années. Dans le cadre d'un tel projet, il apparaît donc très intéressant de proposer des méthodes informatiques susceptibles d'accélérer la découverte d'interactions entre protéines.

Ainsi, au cœur de la révolution actuelle en sciences du vivant, si l'informatique se révèle incontournable pour la production, le stockage et la dissémination de l'information, elle a également la possibilité de jouer un rôle important en ce qui concerne la conception expérimentale en biologie. Il ne s'agit pas uniquement de rendre un service en accomplissant très rapidement une tâche que le biologiste sait accomplir, bien que lentement, il devient envisageable de participer à la réflexion biologique par le biais de l'informatique, en proposant des expériences à réaliser. Il ne s'agit donc plus ici seulement de traiter de l'information : il s'agit de produire de l'information.

1.2 Sujet et approche

C'est dans ce contexte scientifique passionnant et en pleine évolution que se situe mon travail de thèse. La thématique de ce travail concerne les interactions entre protéines. La thèse que je soutiens est que les interactions entre protéines sont en partie explicables et prédictibles à partir des annotations disponibles dans les bases de données publiques, sous réserve que l'on dispose de suffisamment d'exemples de couples de protéines qui interagissent effectivement.

Mon objectif est de contribuer à la découverte d'interactions protéine-protéine, comme suit : je souhaite développer un système de prédictions d'interactions protéine-protéine, afin de pouvoir proposer des expériences à réaliser en priorité et ainsi accélérer le processus d'acquisition de connaissances en biologie. L'approche adoptée est la suivante.

Tout d'abord, en intégrant l'équipe de biologistes dirigée par le professeur Marc Vidal, au Dana Farber Cancer Institute de Boston, j'ai pu participer au projet de cartographie systématique des interactions entre protéines chez *C. elegans* ([Wal00], [Dav01]).

Dans le cadre de ce projet, nous sommes amenés à manipuler de grandes

quantités de données. En particulier, la méthode double hybride par criblage produit des centaines de séquences d'ADN correspondant à des dizaines d'appâts. Ces séquences doivent être alignées sur le génome de *C. elegans* afin d'identifier les interacteurs potentiels. Les traces de séquençage associées doivent être examinées afin de vérifier que les inserts sont dans le bon cadre de lecture. L'ensemble des paramètres expérimentaux et des résultats doit être conservé de manière durable, fiable et standardisée, pour limiter les erreurs humaines liées à la quantité des données traitées et permettre la comparaison et la fusion des résultats provenant de chercheurs différents (travail coopératif). Enfin, comme pour tout projet de génomique fonctionnelle, il est essentiel que la communauté scientifique ait accès aux données produites dans notre laboratoire, dans la mesure où l'analyse des résultats ne peut se faire localement mais doit plutôt être effectuée de manière répartie dans les laboratoires de génétique intéressés par tel ou tel gène.

Ensuite, pour étudier globalement les interactions entre protéines, et au final être capable d'aider la recherche expérimentale en fournissant des prédictions d'interactions, il est nécessaire de disposer du plus grand nombre possible d'exemples, en provenance d'organismes variés, pour pouvoir exploiter la conservation observée des séquences des protéines au cours de l'évolution, ainsi que la conservation soupçonnée de leurs fonctions et de leurs interactions.

Dans cette optique, il est nécessaire de constituer une base de données multi-organismes d'interactions protéine-protéine, fédérant les données issues de notre projet chez *C. elegans* et d'autres données expérimentales provenant de bases de données publiques. L'approche que nous adoptons consiste à représenter chaque protéine par un ensemble de descripteurs. Ces descripteurs doivent être choisis avec soin, dans l'espoir qu'ils puissent expliquer les interactions observées, et donc permettre de caractériser et définir des classes d'interactions entre protéines.

Notre objectif final, comme il a été dit, consiste à exploiter les informations présentes dans cette base de données afin de développer un système de prédiction d'interactions entre protéines. Il s'agit de rechercher des affinités entre descripteurs, sous la forme de règles telle que : une protéine décrite par les descripteurs D1 et D2 est susceptible d'interagir avec toute protéine décrite par les descripteurs D'3, D'4 et D'5. Afin d'induire de telles règles, l'idée est de développer un système d'extraction de connaissances qui repose sur des

algorithmes de Fouille de Données Relationnelles (Datamining), dits de recherche d'ensembles fréquents ([Agr94], [Man94], [Agr96]). Ces algorithmes permettent de rechercher, dans une matrice booléenne où les colonnes sont des traits et les lignes des observations, des ensembles de traits qui sont souvent observés ensemble, c'est-à-dire vrais dans les mêmes lignes.

Comme dans tout processus d'extraction de connaissances, plusieurs tâches doivent être accomplies. En partant d'une base de données constituée avec soin, il est tout d'abord nécessaire de modéliser les données sous une forme appropriée. Il devient alors possible d'appliquer l'algorithme de Datamining choisi. Il est ensuite crucial de concevoir et de mettre en œuvre des filtres de post-traitement, afin de trier, d'évaluer et d'interpréter les règles induites. Enfin, une dernière tâche consiste à exploiter les règles retenues, en l'occurrence dans le cadre d'un logiciel de prédiction d'interactions entre protéines.

Rappelons que l'objectif final de notre travail est de guider les expériences de double hybride réalisées au DFCI dans l'équipe de Marc Vidal, afin d'accélérer la découverte d'interactions. En effet, bien que la détermination de l'interactome, ou ensemble des interactions entre protéines, de *C. elegans* soit l'objectif à long terme de cette équipe, la méthode employée consiste à produire dans un premier temps plusieurs cartes d'interactions de taille moyenne. Typiquement, un chercheur sélectionne 20 à 50 protéines impliquées dans un processus biologique donné, et recherche les interacteurs de ces protéines en employant la technique relativement lourde de "criblage génétique" par double hybride. L'idée est d'exploiter notre système prédictif pour proposer un ensemble d'interacteurs potentiels des protéines d'intérêt. Le collègue biologiste peut alors réaliser une expérience de double hybride "matricielle", bien plus légère, pour tester uniquement les interactions prédites. De cette manière, une partie des interactions concernant les protéines d'intérêt peut être très rapidement découverte.

1.3 Plan de l'exposé

Ce document comporte 6 chapitres. Cette introduction constitue le premier chapitre.

Dans le chapitre 2, nous présentons notre vision de l'état de l'art dans les

domaines de la prédiction d'interactions entre protéines d'une part, et de la fouille de données relationnelles, ou Datamining, d'autre part.

Notre contribution est ensuite présentée dans les chapitres 3, 4 et 5.

Dans le chapitre 3, nous exposons les travaux effectués dans le cadre de deux projets de génomique fonctionnelle, au Massachusetts General Hospital Cancer Center puis au Dana Farber Cancer Institute, au sein de l'équipe du professeur Marc Vidal. Ces projets ont pour objectifs le clonage de l'ensemble des zones codantes de *C. elegans* dans un vecteur qui peut servir de navette vers un grand nombre de vecteurs d'expression divers d'une part, et la cartographie systématique des interactions entre protéines de *C. elegans* à l'aide d'une version améliorée du système double hybride d'autre part. Nous présentons en particulier les outils informatiques développés pour répondre aux besoins de ces projets.

Dans le chapitre 4, nous présentons la base InterDB, une base de données multi-organisme d'interactions protéine-protéine orientée-prédiction que nous avons conçue et construite dans le but de servir de base à un système de prédiction d'interactions entre protéines. InterDB fédère des données provenant de diverses bases de données publiques et privées. Pour annoter les protéines, nous exploitons les bases SwissProt et TrEMBL [Bai99], ainsi que InterPro [Apw01].

Dans le chapitre 5, nous présentons le système d'extraction de connaissances pour la prédiction d'interactions entre protéines que nous avons conçu et mis en œuvre. Ce système exploite les données réunies dans InterDB, et utilise l'algorithme de recherche d'ensembles fréquents *min-ex* [Bou00] pour la partie Datamining. Nous avons eu accès à une implémentation de l'algorithme *min-ex* dans le cadre d'une collaboration avec l'équipe du professeur Jean-François Boulicaut, au sein de laquelle cet algorithme a été développé au Laboratoire d'Ingénierie des Systèmes d'Information de l'INSA-Lyon. Nous présentons en particulier dans ce chapitre les filtres de post-traitement que nous proposons, et qui jouent un rôle clé pour sélectionner les bonnes règles prédictives. Nous terminons ce chapitre par une présentation des résultats obtenus au cours d'une campagne d'évaluation de notre système prédictif, menée à l'aide d'un jeu de validation comportant 113 interactions entre protéines du protéasome de *C. elegans*. Ces données proviennent d'une récente collaboration [Dav01] avec Anne Davy, du Centre de Recherches en Biologie Moléculaire de Montpellier, et avec l'équipe de Marc Vidal.

Nous exposons enfin dans le chapitre 6 les conclusions que nous tirons de ce travail, et considérons les perspectives qui s'offrent à nous pour la suite.

2. Etat de l'art

Nous présentons dans ce chapitre l'état de l'art en ce qui concerne la prédiction d'interactions entre protéines d'une part, et l'extraction de connaissances par recherche d'ensembles fréquents d'autre part.

2.1 Prédiction d'interactions protéine-protéine

Bien que l'importance biologique des interactions entre protéines soit reconnue depuis un certain temps, il apparaît que la bio-informatique ne s'est intéressée que très récemment au problème de la prédiction d'interactions protéine-protéine [Sal99].

Une approche émergente consiste à extraire des informations de la littérature scientifique. L'idée est ici plus de représenter dans un modèle informatique des interactions connues des biologistes, que de prédire de nouvelles interactions. Cette approche repose sur l'observation que les connaissances en biologie sont rarement disponibles sous une forme structurée qui pourrait être exploitée dans un contexte informatique. Par exemple, les résumés d'articles contenus dans la base publique PubMed (http://www.ncbi.nlm.nih.gov/PubMed/), qui recense les publications de 1966 à nos jours dans le domaine de la biologie et de la santé, constituent une mine de connaissances qu'il serait intéressant de pouvoir exploiter de manière automatique.

Dans cette optique, plusieurs groupes de chercheurs s'attachent à extraire des données d'interaction entre protéines à partir des abstracts de PubMed ([Bla99], [Mar01]). Cependant, répétons que cette approche ne peut rien proposer qui ne soit déjà connu des experts biologistes. Ainsi, si elle présente un intérêt indéniable pour la constitution de bases de données d'interactions putatives entre protéines, cette approche ne peut pas réellement être considérée comme une méthode prédictive.

Une autre méthode, proposée indépendamment dans deux articles ([Enr99], [Mar99a]), repose sur l'hypothèse dite des "domaines fusionnés". Cette hypothèse postule que si deux protéines A et B sont homologues à une troisième

protéine mais qu'elles ne sont pas homologues entre elles, alors elles sont susceptibles d'interagir. Cette hypothèse biologique découle d'une interprétation du phénomène de l'évolution des espèces.

Les auteurs de [Mar99a] ont également proposé une autre méthode, basée sur la comparaison de profils phylogénétiques [Pel99]. L'idée est ici que si deux protéines possèdent des homologues dans les mêmes organismes, elles sont fonctionnellement liées et donc susceptibles d'interagir.

Ces deux dernières méthodes sont intégrées dans un système multi-paradigme [Mar99b]. Ce système combine les prédictions fournies par les deux méthodes, et exploite aussi des résultats publiquement disponibles d'expériences de micro-arrays [Eis98], pour proposer des couples de protéines fonctionnellement liées.

Il apparaît donc que très peu de recherches ont été effectuées dans le domaine de la prédiction d'interactions entre protéines. Les quelques méthodes qui ont été proposées récemment reposent généralement sur une hypothèse biologique bien définie, qu'elles valident, implémentent et appliquent.

A notre connaissance, il n'y a eu aucun travail dans la direction que nous suivons (note : excepté les travaux très récents présentés dans [Spr01]), qui consiste à rechercher des associations entre descripteurs protéiques d'origines diverses. En particulier, à notre connaissance aucune publication ne suggère de rechercher des associations entre domaines et familles de protéines répertoriés dans les bases de données comme Pfam [Bat99] ou ProSite [Hof99].

2.2 Techniques d'extraction de connaissances dans les bases de données (KDD)

L'extraction de connaissances dans les bases de données (KDD en anglais, pour Knowledge Discovery in Databases) consiste à découvrir des informations implicites non triviales, précédemment inconnues et potentiellement utiles concernant des données contenues dans des bases de données. Cette extraction est mise en œuvre grâce à un processus semi-automatique et itératif constitué de quatre étapes:

-Sélection des données,

-Préparation des données,

-Recherche de connaissances (en anglais, Datamining),

-Interprétation des résultats.

Cette démarche est suivie au chapitre 5 où chacune de ces étapes est décrite pour l'application visée.

Dans cette section-ci, nous introduisons spécifiquement la technique de Datamining appelée couramment "extraction de règles d'association". C'est celle qui est utilisée au chapitre 5 pour la troisième étape. Après une rapide introduction et une définition du problème, nous exposons l'algorithme de base proposé initialement [Agr94] et qui a été la cible de nombreuses améliorations. Puis, nous présentons un des algorithmes les plus optimisés, en terme de consultation de la base de données, qui a été proposé ultérieurement [Pas00]. Nous en utilisons une variante (voir 5.2.a). Les principaux éléments du présent chapitre sont tirés de [Pas00].

2.2.a Introduction à l'extraction de règles d'association, notion d'itemset fréquent

La notion d'extraction de règles d'association fut introduite par Agrawal et al. dans [Agr93]. L'application type est la suivante: soit une base de données de transactions où chacune d'elles fournit la liste des articles achetés par un client

dans un grand magasin, déterminer les règles signalant les cas où la présence de certains d'entre ces articles entraîne significativement la présence d'autres. Plus précisément, une règle d'association est de la forme X -> Y, où X et Y sont des ensembles d'articles et s'interprète de la façon suivante: les transactions qui contiennent les articles de l'ensemble X ont une tendance significative à contenir aussi les articles de l'ensemble Y.

Exemple :

Une telle règle peut être: "Les transactions contenant les articles pain et vin contiennent aussi l'article fromage".

On associe à une règle d'association deux mesures:

- la confiance qui indique la proportion de transactions contenant à la fois les articles de X et ceux de Y parmi celles contenant les articles de X;

- la fréquence qui indique la proportion de transactions contenant à la fois les articles de X et ceux de Y parmi toutes les transactions.

Exemple :

Si la règle précédente possède une confiance de 40% et une fréquence de 5%, ceci veut dire qu'elle est respectée dans 40% des cas possibles (ceux où il y a eu achat de pain et de vin) et dans 5% de tous les cas.

L'enjeu consiste à trouver, avec de bonnes performances en temps et en espace, l'ensemble des règles d'association dont la confiance et la fréquence sont supérieurs à des seuils laissés à l'appréciation de l'utilisateur.

Il apparaît que cette approche a été fructueuse dans de nombreux domaines comme la planification commerciale, l'aide au diagnostic en recherche médicale, l'organisation et l'accès à des sites Internet ou l'analyse d'images. Nous l'appliquons ici pour un problème où beaucoup de données sont en jeu, en terme de nombre de transactions et surtout en terme de nombre d'articles.

2.2.b Définition du problème

Le problème peut être formalisé de la façon suivante ([Agr93], [Agr94]) :

Soient :

- $I = \{i_1,..., i_m\}$ un ensemble de m articles ou "items"

- $B = \{t_1,..., t_n\}$ une base de données de n transactions ou "objets", où chaque transaction t_i est un sous-ensemble de I.

La fréquence fréquence(I) d'un sous-ensemble I ou "itemset" de *I* est défini par:

fréquence(I) = $|\{ t \in B \mid I \subseteq t \}| / |\{ t \in B \}|$

Un itemset dont la fréquence est supérieur au seuil fixé par l'utilisateur est dit itemset "fréquent". Nous dirons dans la suite souvent simplement ensemble fréquent au lieu d'itemset fréquent.

Une règle d'association r est de la forme $I_1 \rightarrow I_2$ où I_1 et I_2 sont des itemsets et tels que $I_1 \cap I_2 = \varnothing$.

La confiance confiance(r) d'une règle r est définie par:

confiance(r) = fréquence($I_1 \cup I_2$) / fréquence(I_1)

La fréquence fréquence(r) d'une règle r est défini par:

fréquence(r) = fréquence($I_1 \cup I_2$)

L'objectif de l'extraction de règles d'association d'une base de données *B* est la détermination des règles dont la fréquence et la confiance sont au moins égaux à des seuils minimaux de fréquence minfréquence et de confiance minconfiance. Ce problème se décompose en deux sous-problèmes:

1. La détermination des ensembles fréquents de *B*, c'est à dire des itemsets dont la fréquence est supérieur ou égal à minfréquence.

2. La détermination des règles d'association de la forme I' -> I - I', où I est un ensemble fréquent et I' tel que I' $\subset$ I, dont la confiance est supérieure ou égale à minconfiance.

On note tout de suite deux difficultés algorithmiques:

- Le premier problème présente une complexité exponentielle dans la mesure où, si l'on considère un ensemble d'items *I* de taille m, le nombre potentiel d'ensembles fréquents est 2^m.

- Le second problème présente aussi une complexité exponentielle car pour un ensemble fréquent I donné le nombre potentiel de règles d'association est $2^{|I|} - 2$.

On peut remarquer cependant que, de ce point de vue, la difficulté majeure réside dans le calcul des ensemble fréquents en particulier dans la mesure où il impose de parcourir la base de données qui peut contenir de nombreux objets.

On note que, dans notre application, les règles à apprendre sont telles que la conclusion est toujours la même: il s'agit de l'article "interaction". En effet, nous cherchons à découvrir des règles donnant les prémisses d'une interaction entre deux protéines. D'une certaine façon, on peut dire que l'apprentissage que nous devons pratiquer est supervisé. Dans ce cadre, ce sont donc uniquement les algorithmes de recherche d'ensembles fréquents qui nous intéressent.

L'ensemble F des ensembles fréquents est défini par:

$F = \{ \ i \subseteq I \ | \ i \neq \varnothing \wedge \text{fréquence}(i) \geq \text{minfréquence} \}$

Exemple, tiré de [Pas00] et qui est utilisé pour illustrer les développements de ce chapitre:

Soient:

I = {A, B, C, D, E}

B = {{ A, C, D},

{B, C, E},

{A, B, C, E},

{B, E},

{A, B, C, E},

{B, C, E}}

minfréquence = 2/6

Les ensembles fréquents de taille 2 portant sur les items A, B et C sont les suivants:

{A, B} de fréquence 2/6

{A, C} de fréquence 3/6

{B, C} de fréquence 4/6

Nous noterons dans la suite souvent les ensembles sous la forme de chaîne de caractères, par exemple ABD au lieu de {A, B, D}.

2.2.c L'algorithme Apriori

Cet algorithme est à la base de nombreuses versions améliorant ses performances. Il repose sur le fait que les ensembles fréquents potentiels forment un treillis possédant des propriétés intéressantes permettant d'éviter une énumération exhaustive des ensembles fréquents potentiels, c'est à dire des sous-

ensembles de *I*.

2.2.c.1 Treillis des ensembles fréquents potentiels

Ce treillis est formé en considérant l'ensemble des parties de *I* (i.e. l'ensemble des ensembles fréquents potentiels) avec comme relation d'ordre l'inclusion.

Exemple:

Pour *I* = {A, B, C, D, E}, ce treillis peut être représenté par le schéma suivant (voir figure 2.1).

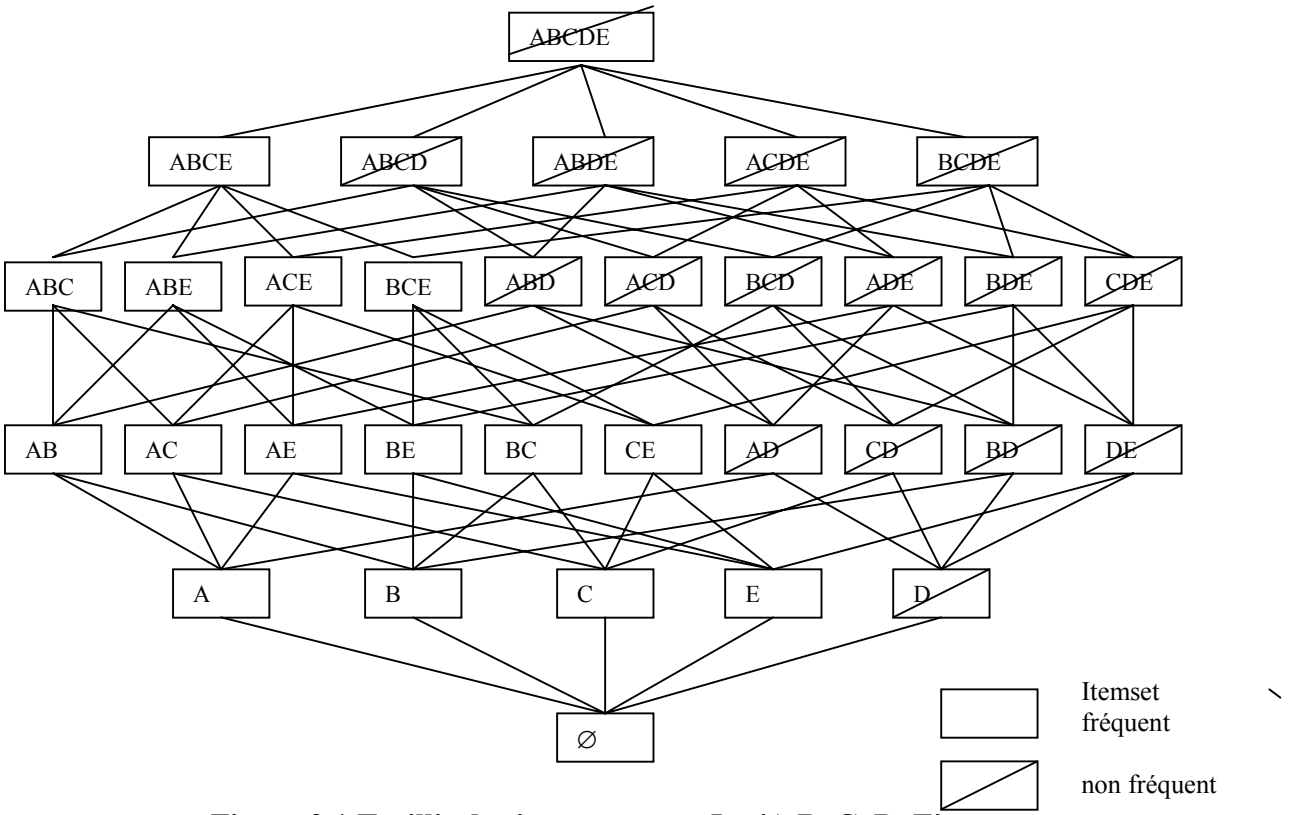

Figure 2.1 Treillis des itemsets pour *I* = {A,B, C, D, E}.

On trouve aussi sur ce schéma les ensembles fréquents avec *B* selon l'exemple considéré: les ensembles non fréquents sont marqués par la diagonale.

Les deux propriétés exploitées par l'algorithme Apriori concernant les ensembles fréquents présents dans un tel treillis sont complémentaires. Ce sont les

suivantes:

Propriété 1.1: Tous les sous-ensembles d'un ensemble fréquent sont fréquents.

Cette propriété permet à l'algorithme Apriori de limiter la construction des ensembles "candidats", c'est à dire susceptibles d'être fréquents, de taille k à partir des ensembles fréquents de taille k-1.

Propriété 1.2: Tous les sur-ensembles d'un ensemble non fréquent sont non fréquents.

Cette propriété permet à l'algorithme Apriori d'éliminer un ensemble candidat si un seul de ses sous-ensembles est non fréquent.

2.2.c.2 Programme Apriori

L'algorithme Apriori peut être décrit par le programme suivant (Figure 2.2) qui admet en données *I*, *B* et minfréquence et qui fournit en résultat les ensembles fréquents :

début

% F_k est l'ensemble des ensembles fréquents de taille k. Un ensemble fréquent est représenté par une structure possédant deux champs: itemset (les items de l'ensemble) et fréquence.

C_k est l'ensemble des ensembles candidats de taille k issu de F_{k-1} . Ces ensembles candidats respectent les propriétés 1.1 et 1.2. Un ensemble candidat est représenté aussi par une structure possédant deux champs: itemset (les items de l'ensemble) et fréquence.

F_k est l'ensemble des ensembles fréquents de taille k. Un ensemble fréquent est représenté de la même façon qu' un ensemble candidat.

L'union des ensembles F_k est le résultat attendu. %

F_1 : = ensembles_fréquents1(*I*, *B*) ;

k := 2 ;

tantque $F_{k-1} \neq \varnothing$

faire C_k := Apriori-Gen(F_{k-1}) ;

pour chaque $t \in B$

faire S_t := Subset(C_k, t) ;

% S_t est l'ensemble des ensembles candidats de C_k contenus dans la transaction t%

pour chaque $s \in S_t$ faire s.fréquence := s.fréquence + 1 fait

fait ;

F_k : = {$c \in C_k$ | c.fréquence $\geq$ minfréquence} ;

k := k + 1

fait
fin

Figure 2.2 : Extraction des ensembles fréquents à l'aide du programme Apriori.

Ce programme fait usage de la procédure Apriori-Gen(F_{k-1}) qui fournit l'ensemble des ensembles candidats C_k de taille k à partir de F_{k-1} . Cette procédure procède comme suit (Figure 2.3).

début
% l'ensemble C est le résultat attendu (l'ensemble des ensemble candidats de taille k) %
% La Propriété 1.1 est utilisée pour initialiser C.
On note qu'un ordre total < est introduit sur les items pour initialiser rapidement C (il est utile aussi pour la représentation des ensembles). Ce peut être par exemple l'ordre lexicographique.
Aussi, dans cette description , il est fait abstraction des champs fréquence des éléments de F_{k-1} et de C (dont la valeur est 0) %
C := {{ p_1, ..., p_{k-1}, q_{k-1}} | { p_1, ..., p_{k-1}}$\in$ F_{k-1} , { q_1, ..., q_{k-1}}$\in F_{k-1,}$
$p_1 = q_1$, ..., $p_{k-2} = q_{k-2}$, $p_{k-1} < q_{k-1}$ };
% La Propriété 1.2 est utilisée pour éliminer de C des candidats non fréquents %
pour chaque c $\in$ C
faire pour chaque s $\subset$ c, |s| = k-1
faire si s $\notin$ F_{k-1} alors C := C-{s} fsi fait
fait ;
C
fin

Figure 2.3 : Production des ensembles candidats de taille k avec Apriori-Gen

Les autres procédures, ensembles_fréquents1(*I*, *B*) et Subset(C_k, t) ne posent pas de problèmes particuliers de compréhension.

Exemple d'exécution de Apriori:

Les étapes suivantes sont franchies:

- C_1 = { (A, 3/6), (B, 5/6), (C, 5/6), (D, 1/6), (E, 5/6) }
  F_1 = { (A, 3/6), (B, 5/6), (C, 5/6), (E, 5/6) }

-1ère étape de l'itération :

initialisation de C_2 : = { (AB,0), (AC,0), (AE, 0), (BC, 0), (BE, 0), (CE, 0) }

obtention de C_2 = { (AB, 2/6), (AC, 3/6), (AE, 2/6), (BC, 4/6), (BE, 5/6), (CE, 4/6) }

F_2 = { (AB, 2/6), (AC, 3/6), (AE, 2/6), (BC, 4/6), (BE, 5/6), (CE, 4/6) }

-2ème étape de l'itération :

initialisation de C_3 : = { (ABC, 0), (ABE, 0), (ACE, 0), (BCE, 0) }

obtention de C_3 = { (ABC, 2/6), (ABE, 2/6), (ACE, 2/6), (BCE, 4/6) }

F_3 = { (ABC, 2/6), (ABE, 2/6), (ACE, 2/6), (BCE, 4/6) }

-3ème étape de l'itération :

initialisation de C_4 : = { (ABCE,0) }

obtention de C_4 = { (ABCE, 2/6) }

F_4 = { (ABCE, 2/6) }

2.2.d L'algorithme Close, notion d'itemset fermé.

On note que l'itération interne du programme Apriori, qui procède à la construction des ensembles $C_{k,}$ parcourt l'ensemble *B* des objets. Elle est coûteuse dans la mesure où elle est interne et dans la mesure où elle conduit à des accès à la base de données contenant *B*. De nombreux développements ont conduit à une amélioration de l'algorithme Apriori, essentiellement afin de diminuer le coût de ces parcours. Nous présentons ici un de ces développement particulièrement intéressant de ce point de vue, l'algorithme Close. Cet algorithme opère en quelque sorte une abstraction de *B* en procédant à une extraction d'ensembles fréquents dit "fermés" à partir desquels il est possible de déduire tous les ensembles fréquents.

Nous exposons rapidement d'abord la notion de fermeture utilisée, dite fermeture de la connexion de Galois, puis l'extension de la notion de treillis et les propriétés associées aux ensembles fréquents fermés et enfin l'algorithme lui-même.

2.2.d.1 Connexion de Galois

Cette notion est définie comme le couple des deux applications suivantes ϕ

et ψ que nous présentons informellement de la façon suivante:

- ϕ fournit l'ensemble des éléments de *I* communs aux objets de *B* donnés en paramètre.

Exemple: ϕ ({ACD, BCE}) = {C}

- ψ fournit l'ensemble des objets de *B* contenant tous les items de *I* donnés en paramètre.

Exemple: ψ ({A, C}) = {ACD, ABCE, ABCE}

L'opérateur de fermeture de la connexion de Galois utilisé est $\gamma = \phi \circ \psi$ où $\phi \circ \psi$ (i) = $\phi(\psi(i))$ pour i $\subseteq$ *I*.

Un itemset i $\subseteq$ *I* est dit "fermé" si γ (i) = i. Intuitivement, un itemset est fermé si aucun autre item n'est commun à l'ensemble des transactions de *B* qui contiennent cet itemset.

Exemple: L'ensemble BCE est fermé car:

ψ({B, C, E}) = {BCE, ABCE, ABCE, BCE}

ϕ({BCE, ABCE, ABCE, BCE}) = {B, C, E}

L'ensemble BC n'est pas fermé car tous les objets de *B* contenant B et C (c'est à dire BCE, ABCE, ABCE, BCE) ont aussi E en commun.

On note que γ(i) fournit le plus petit ensemble fermé (au sens de l'inclusion) contenant i.

Exemple: γ(BC) = BCE

2.2.d.2 Treillis des ensembles fermés

Ce treillis est formé d'une façon analogue à celui des itemsets. Il porte sur les itemsets fermés.

Exemple: on trouve dans la Figure 2.4 ce treillis (en caractère gras) superposé au treillis des itemsets.

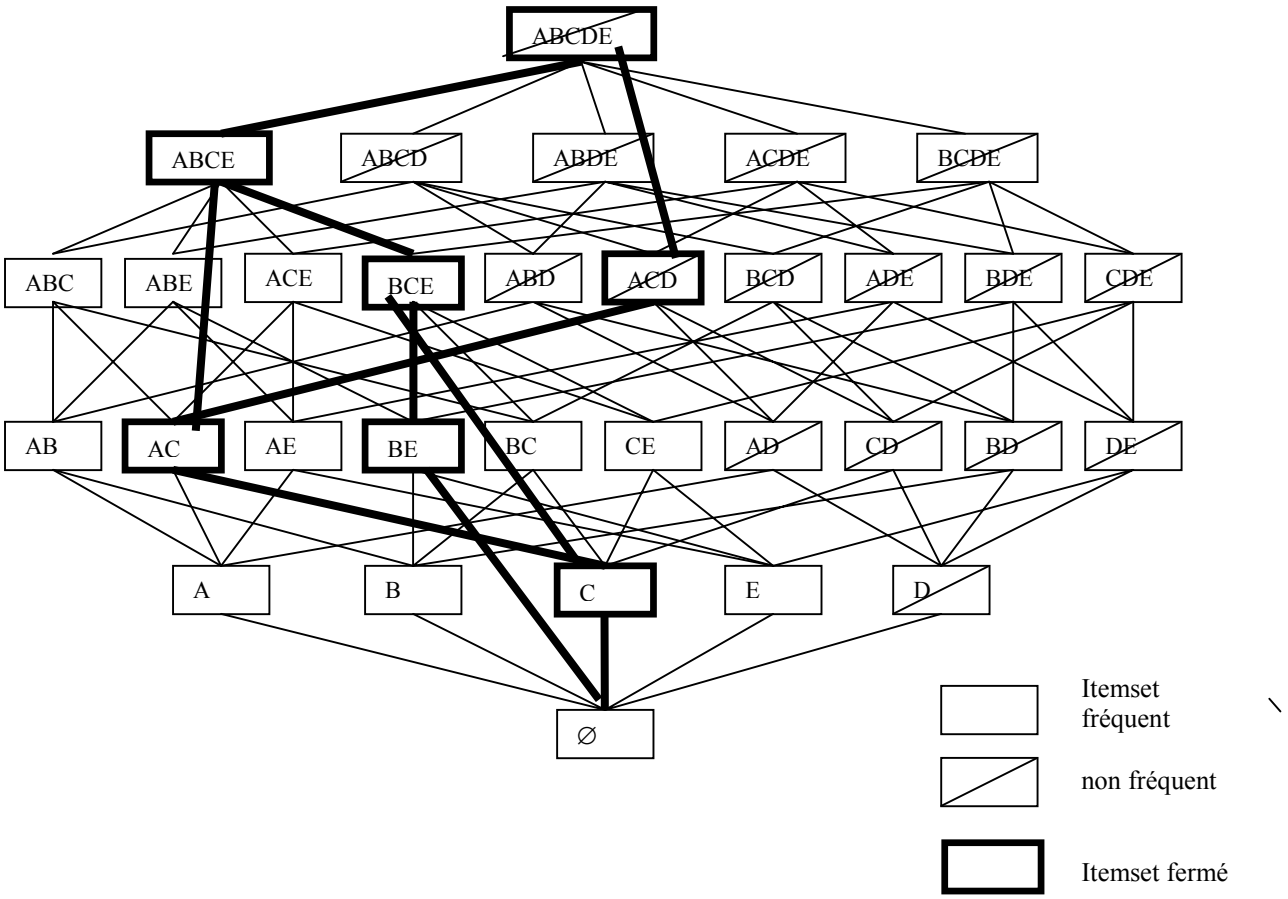

Figure 2.4 : Treillis des ensembles fermés relatif à *B*.

De même, la notion d'ensemble fermé fréquent est une extension de celle d'ensemble fréquent.

Exemple: BCE est fréquent car

fréquence(BCE) = | ψ({B, C, E}) | / | *B* | = 4/6 > minfréquence.

On définit aussi par extension l'ensemble des ensembles fermés fréquents FF et aussi l'ensemble des ensembles fermés fréquents "maximaux" FM:

FM = { i $\subseteq I$ | i $\in$ FF $\wedge$ ($\forall$ i' $\supset$ i, fréquence(i') < minfréquence)}

Exemple: la figure 2.4 illustre bien que FM = {ABCE}

Les propriété 1.1 et 1.2 sont évidemment applicables aux ensembles fréquents. De plus, deux autres propriétés leur sont propres:

Propriétés 1.3: la fréquence d'un itemset i est égal à la fréquence de sa fermeture, soit:

$\forall$ i $\subseteq I$, fréquence(i) = fréquence(γ (i))

Exemple: la figure 2.4 illustre bien que support(AB) = support(ABC) = support(ABCE) = 2/6.

Propriété 1.4: l'ensemble des ensembles fermés fréquents maximaux est le même que l'ensemble des ensembles fréquents maximaux.

On en déduit une propriété essentielle qui est utilisée par l'algorithme Close:

Propriété 1.5: L'ensemble des ensembles fermés fréquents est une "base" pour l'ensemble des ensembles fréquents, c'est à dire que tout ensemble fréquent est sous-ensemble d'un ensemble fermé fréquent.

Exemple: la Propriété 1.5 est illustrée grâce à la figure 2.4 en observant que tous les ensembles fréquents sont des sous-ensembles de l'ensemble fermé fréquent maximal ABCE.

2.2.d.3 L'Algorithme Close

L'approche est fondée sur l'idée suivante:

-1ère étape: détermination de FF

2ème étape: déduire les ensembles fréquents à partir de FF par production de tous les sous-ensembles des éléments de FF et déterminer leur support à partir de leur fermeture.

C'est la première étape qui est cruciale. Ses performances doivent être meilleures pour l'extraction des ensembles fermés fréquents qu'Apriori pour l'extraction des ensemble fréquents. L'idée essentielle est d'utiliser un critère supplémentaire par rapport à Apriori pour réduire encore plus les itemsets fréquents candidats (dans ce cas, ils sont aussi fermés).

Intuitivement, n'est plus aussi considéré comme candidat un k+1-itemset g

construit à partir d'un k-itemset fréquent s (c'est à dire tel que $s \subset g$) si $g \subset \gamma(s)$. Toujours intuitivement, cela se comprend comme suit :

- D'une part, $\gamma(g) = \gamma(s)$. En effet, on peut montrer que pour tout itemset e, $\gamma(e)$ est le plus grand sur-ensemble de e ayant même fréquence que e. Or, on a $s \subset g \subset \gamma(s)$, donc fréquence($\gamma(s)$) ≤ fréquence(g) ≤ fréquence(s). Etant donné que fréquence(s) = fréquence($\gamma(s)$), il ressort que fréquence($\gamma(s)$) = fréquence(g) : $\gamma(s)$ est un sur-ensemble de g de même fréquence que g. Un simple raisonnement par l'absurde permet alors d'établir que $\gamma(s)$ est le plus grand sur-ensemble de g de même fréquence que g, soit $\gamma(g) = \gamma(s)$. Finalement, il n'est donc pas utile de déterminer le fermé de g.

- D'autre part, il n'est pas utile de considérer g pour produire les k+1 générateurs. En effet, les fermés de tels k+1 générateurs seraient les mêmes que ceux obtenus à partir de s. Ce constat repose sur le constat suivant : la fermeture de tout ensemble $g \cup h$, pour tout h, peut être aussi obtenue à partir de s, où g est un sur-ensemble de s contenu dans la fermeture de s. On le démontre grâce à la propriété suivante sur les fermetures d'union d'ensembles:

$\forall$ (i1,i2), $\gamma(i1 \cup i2) = \gamma(\gamma(i1) \cup \gamma(i2))$

En effet, $\gamma(g \cup h) = \gamma(\gamma(g) \cup \gamma(h)) = \gamma(\gamma(s) \cup \gamma(h))) = \gamma(s \cup h)$.

Exemple: C'est le cas (voir Fig. 1.4) pour g = ABC, s = AB et $\gamma(s)$ = ABCE. L'ensemble ABC n'est pas considéré comme candidat.

2.2.d.4 Programme Close

La mise en œuvre de Close utilise la notion d'itemset "générateur" pour produire les candidats. Un itemset générateur d'un itemset fermé f est un itemset minimal (au sens de l'inclusion) dont la fermeture est f.

Exemple: L'ensemble d'itemsets générateurs de l'itemset fermé BE est {B, E} car $\gamma(E) = \gamma(B) = \gamma(BE)$.

L'algorithme Close peut être décrit par le programme suivant (Figure 2.5).

début

%Un k-générateur est une structure à trois champs: générateur (un k-itemset), fermeture (celle du générateur), fréquence (celle du générateur, identique à celle de la fermeture).

FF_k est l'ensemble des k-générateurs fréquents.

FFC_k est l'ensemble des k-générateurs candidats issu de FF_{k-1} .

La notation FFC_k .générateurs désigne l'ensemble des itemsets générateur des k-générateurs de FFC.

L'union des ensembles FF_k est le résultat attendu. %

FFC_1. générateurs : = *I* ;

k := 1 ;

tantque FFC_k. générateurs $\neq \varnothing$

faire FFC_k := Gen-Closure(FFC_k, *B*) ;

% les champs fermeture et support des éléments de FFC_k sont mis à jour %

$FF_k := \varnothing$;

pour chaque $c \in FFC_k$

faire si c.fréquence $\geq$ minfréquence alors $FF_k := FF_k \cup \{c\}$ fsi fait;

FFC_{k+1}. générateurs : = Gen-Generator(FF_k) ;

% les champs générateur des éléments de FFC_{k+1} sont mis à jour à partir de FF_k en utilisant les propriétés 1.1et 1.2 et le critère cité en 2.2.d.3 %

k := k + 1

fait

fin

Figure 2.5 : Extraction des ensembles fermés fréquents à l'aide du programme Close

Exemple d'exécution de Close:

Les étapes suivantes sont franchies:

1ère étape:

FFC_1 = { (A, AC, 3/6), (B, BE, 5/6), (C, C, 5/6), (D, ACD, 1/6), (E, BE, 5/6) }

FF_1 = { (A, AC, 3/6), (B, BE, 5/6), (C, C, 5/6), (E, BE, 5/6) }

2ème étape:

FFC_2 = { (AB, ABCE, 2/6), (AE, ABCE, 2/6), (BC, BCE, 4/6), (CE, BCE, 4/6) }

FF_2 = { (AB, ABCE, 2/6), (AE, ABCE, 2/6), (BC, BCE, 4/6), (CE, BCE, 4/6) }

On note qu'à l'issue cette étape, aucun candidat ne peut être construit à partir de FF_2 à cause du critère introduit en 2.2.d.3.

3. Génomique fonctionnelle chez *C. elegans*

Le nématode *Caenorhabditis elegans* est un organisme multicellulaire particulièrement agréable à manipuler : il est hermaphrodite auto-fécondant, congelable, translucide, et bénéficie d'un cycle de reproduction court. Ces caractéristiques en font un organisme de choix en génétique, notamment pour l'étude du développement et du comportement ([Ahr97], [Kuw97], [Wal98]). Il a servi d'organisme modèle pour les projets de grand séquençage, et son génome est le premier d'un organisme multicellulaire à avoir été intégralement séquencé [Cel98]. Depuis la publication de son génome en 1998, plusieurs projets de génomique fonctionnelle concernant *C. elegans* ont vu le jour, allant de l'amputation systématique de zones codantes [KOC], l'élimination fonctionnelle de gènes par interférence ARN ([Gon00], [Fra00], [Pia00], [Mae01]), jusqu'au développement de micro-arrays globaux ([Rei00], [Hil00], [Jia01]), en passant par la cartographie des interactions protéine-protéine [Wal00].

Dans une première partie, l'approche que nous avons développée et mise en œuvre pour évaluer la qualité des prédictions de gènes chez C. elegans est exposée [Reb01]. Cette approche repose sur l'amplification par PCR à partir d'une banque de cDNAs, puis le clonage à l'aide du système Gateway [Wal00b] et le séquençage, de gènes caractérisés ou prédits. Les gènes ainsi clonés représentent une partie de l'ORFéome (ensemble des zones codantes) de C. elegans, qui constitue une ressource très utile pour des projets de génomique fonctionnelle. La seconde partie présente notre projet de cartographie des interactions protéine-protéine chez C. elegans par double hybride, en cours dans le laboratoire du professeur Marc Vidal au Dana Farber Cancer Institute. L'exposé décrit en particulier le projet pilote de cartographie des interactions entre protéines impliquées dans le développement de la vulve [Wal00], en insistant sur les aspects bioinformatiques.

3.1 Clonage de l'ORFéome

Lorsque la séquence du génome d'un organisme est disponible, la génomique fonctionnelle peut proposer des annotations fonctionnelles pour les zones codantes de ce génome, en appliquant des techniques expérimentales à grande échelle. Pour ce faire, deux problèmes doivent tout d'abord être résolus. Il s'agit de l'identification des zones codantes, et du clonage dans divers vecteurs d'expression de l'ensemble de ces zones codantes.

Nous présentons dans cette partie une approche que nous avons mise en œuvre au Dana Farber Cancer Institute pour tenter de répondre simultanément à ces deux problèmes [Reb01]. Cette stratégie est tout d'abord décrite, puis les principaux résultats sont exposés.

3.1.a Objectif et méthode

Depuis 1998, nous disposons grâce au projet génome de la séquence complète du génome de *Caenorhabditis elegans*. Le génome contient l'intégralité de l'information biologique. Son déchiffrage était donc une première étape indispensable, et il constitue en soi une ressource fantastique. La seconde étape consiste à identifier les gènes. Cependant la séquence génomique brute s'avère difficile à décoder, surtout chez les eucaryotes où des choix alternatifs de promoteurs, d'épissage et de polyadenylation, ainsi que la capacité à éditer le RNA (modifications post-transcriptionnelles) génèrent de nombreux variants de l'information génomique.

Une possibilité consiste à s'appuyer sur les prédictions de gènes disponibles pour l'organisme étudié. Cependant, même si les prédictions fournissent de bonnes indications, elles ne doivent néanmoins pas être prises pour argent comptant. En effet, la qualité des prédictions dépend, entre autres, de l'organisme, des logiciels et méthodes employés, et des exigences ou objectifs que se fixent les responsables de l'annotation. Il peut par exemple sembler biologiquement paradoxal que *C. elegans* (19000 gènes prédits, [Cel98]), possède

moitié plus de gènes que la mouche *Drosophila melanogaster* (13600 gènes prédits, [Ada00]). Il est en tous cas clair que le problème de la prédiction *ab initio* de zones codantes chez les eucaryotes reste à ce jour un problème difficile et ouvert. Les prédictions de gènes qui n'ont pas encore été validées par des résultats expérimentaux doivent donc être confirmées. C'est l'objectif du projet transcriptome chez *C. elegans* [Thi01b], qui définit les gènes à partir de l'alignement de cDNAs sur le génome.

Il s'agit ensuite de déterminer les fonctions des gènes identifiés. Dans cette optique, la génomique fonctionnelle peut fournir très rapidement des indices, simplifiant ainsi le travail du généticien. Or en général les projets de génomique fonctionnelle vont nécessiter le clonage, dans divers vecteurs d'expression, de l'ensemble des zones codantes, ou ORFéome (les ORFs pour Open Reading Frames désignent les séquences codantes).

Afin d'attaquer simultanément ces deux problèmes (identification et clonage des gènes) chez *C. elegans*, nous appliquons la stratégie suivante. Des amorces de PCR sont conçues au vu des séquences de gènes prédits, à l'aide du logiciel OSP [Hil91]. Ces amorces sont utilisées pour amplifier des ORFs par PCR à partir d'une banque hautement représentative de cDNAs de *C. elegans*. Ces ORFs sont ensuite clonés à l'aide du système de clonage par recombinaison Gateway [Wal00b], basé sur les réactions de recombinaison mises en œuvre par le phage lambda pour son intégration et excision du génome de *E. coli*. Cette technique permet le clonage directionnel de produits de PCR dans un vecteur de référence, puis le transfert des inserts résultants vers de nombreux vecteurs d'expression *in vitro*, sans nécessiter l'utilisation d'enzymes de restriction ni de ligases. Enfin, les inserts sont séquencés et alignés sur le génome de *C. elegans*, afin de confirmer ou de corriger les prédictions initiales.

3.1.b Mise en œuvre

En pratique, des scripts en Perl ont été écrits pour automatiser complètement l'étape de conception d'amorces de PCR. Ces scripts communiquent avec une copie locale de la base de données publique WormBase ([Ste01b], http://www.wormbase.org), qui regroupe l'ensemble des informations (génétiques,

génomiques, bibliographiques) concernant *Caenorhabditis elegans*.

La base WormBase utilise le système de gestion de bases de données orienté objet Acedb [Acedb], [Dur94], [Thi99]. Le système Acedb a été initialement développé pour gérer et distribuer les données génétiques et génomiques de *C. elegans*. Son utilisation s'est depuis généralisée, en particulier dans le domaine des sciences du vivant. Il est utilisé notamment dans plusieurs grands centres de séquençage dans le cadre du Human Genome Project. Pour communiquer avec notre copie locale de WormBase, nous tirons parti du système AcePerl [Ste99]. Il s'agit d'une interface à Acedb pour la programmation en langage Perl.

Le protocole pour la conception des amorces de PCR est le suivant.

Tout d'abord, la liste des noms des gènes ciblés est générée. Cette liste peut provenir d'une source extérieure ou d'une saisie manuelle, ou bien être obtenue par un ensemble de requêtes Acedb dans WormBase, ou encore par un script Perl en fonction de critères généraux. Par exemple, pour estimer le nombre de gènes de *C. elegans* (voir 3.1.c), nous avons sélectionné aléatoirement un échantillon de 1222 gènes prédits ne comportant aucune confirmation expérimentale de type EST.

Ensuite, pour chaque gène de cette liste, notre logiciel recherche dans WormBase la séquence codante correspondante (ORF). Les ORFs ne débutant pas par un codon START ou ne finissant pas par un codon STOP sont écartés. La recherche d'amorces proprement dite est alors effectuée, à l'aide du programme OSP [Hil91]. Ce programme, utilisé dans plusieurs grands centres de séquençage, prend en compte notamment le risque d'hybridation de chaque amorce avec elle-même et avec l'autre amorce, et peut rejeter des amorces peu spécifiques au vu de la séquence génomique complète. Nous lui imposons des contraintes sur la longueur des amorces à générer et sur la température de fusion (TM) souhaitée. Ces contraintes ont été optimisées pour répondre aux objectifs et aux conditions expérimentales mis en œuvre dans notre laboratoire.

Ensuite, les amorces proposées sont examinées, et lorsque plusieurs amorces sont identiques un seul exemplaire est retenu. Cette situation peut correspondre par exemple à des formes alternatives d'un gène unique. Ou bien, les amorces peuvent provenir de deux soumissions indépendantes de la même séquence codante. En effet, comme la plupart des bases de données biologiques et

en dépit des efforts des annotateurs, WormBase contient certaines informations redondantes. Finalement, des séquences communes nécessaires au clonage des produits de PCR dans le système Gateway (voir ci-dessous) sont ajoutées en tête des amorces retenues. Le formulaire de commande est alors présenté à l'utilisateur pour accord et envoi à une entreprise privée, qui synthétise nos amorces.

Pour faciliter l'analyse, les réactions de PCR sont organisées par ordre de taille prédite des produits croissante (figure 3.1.1). Il est ainsi très simple d'identifier, au vu du gel d'agarose, les séquences amplifiées dont la taille ne correspond pas à celle de la séquence codante prédite associée. Dans ces cas, il

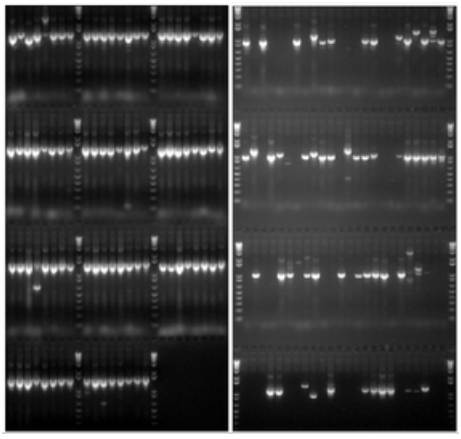

Figure 3.1.1 : photographies de gels de PCR

Les produits sont rangés par taille prédite croissante, afin de faciliter l'identification des séquences qui diffèrent des prédictions, et de permettre l'optimisation des conditions de PCR.

A gauche, amplification de gènes pour lesquels une confirmation complète a été obtenue par le projet de séquençage de cDNAs (Y. Kohara, voir 3.1.c). La grande majorité des produits a la taille attendue.

A droite, amplifications de gènes prédits mais ne disposant d'aucune confirmation expérimentale (aucun cDNA). Environ 1/3 des réactions n'ont rien amplifié, et 1/5 des produits ne sont pas de la taille prévue.

peut s'agir soit d'erreurs de prédiction, et alors nous pourrons corriger les prédictions erronées après séquençage; soit d'artéfacts de PCR qui doivent absolument être repérés et écartés de l'expérience. De toute manière, ces séquences devront être analysées avec une attention particulière (figure 3.1.2). Par

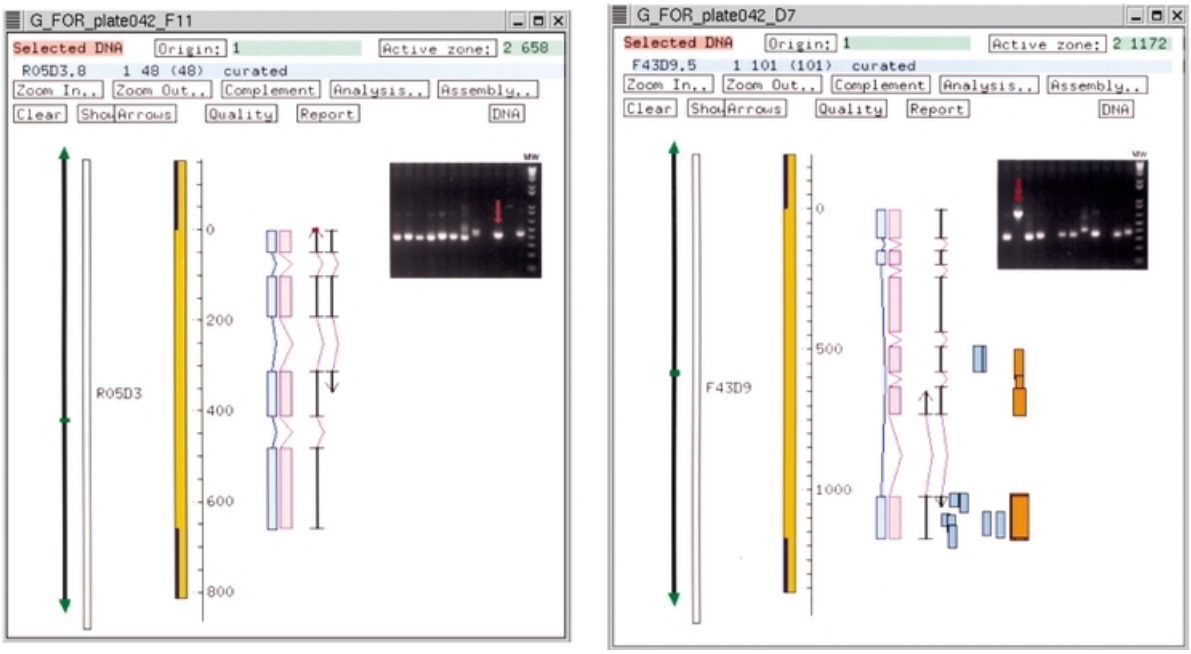

Figure 3.1.2 : Alignement d'OSTs sur le génome de *C. elegans* et correction des prédictions initiales

Cette figure présente l'interface du logiciel Acembly pour les OSTs correspondant aux gènes prédits R05D3.8 et F43D9.5, respectivement à gauche et à droite. La séquence génomique est vue de haut en bas. Les boites bleues (exons) reliées par des traits bleus (introns) représentent le gène prédit. Les boites roses et traits roses représentent la reconstruction du gène réel, construit à partir des flèches noires qui représentent les OSTs. Chaque gène cloné est séquencé dans les deux sens (5 prime vers 3 prime et inversement), d'où les deux flèches. En haut à droite de chaque image figure le gel de PCR, où une flèche rouge désigne le produit examiné.

Pour le gène R05D3.8 (à gauche), le produit de PCR a la bonne taille, et le séquençage confirme que la prédiction est correcte.

Au contraire, le produit de PCR est visiblement trop long pour F43D9.5 (image de droite). Dans ce cas il ne s'agit pas d'un artéfact expérimental, l'OST s'aligne au bon endroit dans le génome mais la prédiction a manqué trois exons intermédiaires. Nous pouvons donc corriger la prédiction pour ce gène. En l'occurrence, la nouvelle structure du gène permet d'identifier plusieurs homologues potentiels de ce gène par utilisation de Blast, alors qu'un Blast avec la séquence prédite initiale ne produit aucune similarité significative.

ailleurs, cette organisation par taille prédite croissante nous permet d'adapter les conditions de PCR à la taille des fragments que nous cherchons à amplifier. En particulier, les durées d'élongation sont optimisées pour chaque plaque 96-puits.

Les fragments amplifiés avec succès sont clonés dans un vecteur de

référence à l'aide du système Gateway. Ce système est basé sur les réactions de recombinaison mises en œuvre pour l'intégration et l'excision du phage lambda dans le génome de *E. coli*. Il s'agit de réactions enzymatiques de recombinaison, assurant une haute fidélité et une excellente efficacité. D'une part, le système utilise la réaction d'intégration pour permettre de cloner de manière directionnelle des produits de PCR dans le vecteur de référence Gateway. Pour ce faire, il est nécessaire que les extrémités de ces fragments contiennent des séquences spécifiques, d'une vingtaine de paires de bases. Ce sont ces séquences qui sont rajoutées en tête de nos amorces de PCR. D'autre part, en s'inspirant de la réaction d'excision, le système Gateway permet de transférer les inserts clonés dans le vecteur de référence vers divers vecteurs de destination. Le vecteur de référence joue donc un rôle de navette, et les séquences ainsi clonées peuvent être utilisées pour une grande variété d'expériences, en fonction du vecteur de destination choisi.

Les inserts clonés avec succès sont alors séquencés, afin de déterminer précisément les structures des gènes correspondants. Les séquences obtenues, appelées OSTs (pour ORF Sequence Tags), correspondent en effet à des ORFs provenant d'une banque de cDNAs, et ne contiennent donc que des exons. En pratique, les OSTs sont alignées sur le génome à l'aide du logiciel Acembly d'assemblage et d'édition de séquences [Acembly]. L'interface conviviale et puissante de ce logiciel permet d'éditer les traces de séquençage au vu de la séquence génomique correspondante, pour déterminer de façon exacte les frontières entre introns et exons. Nous pouvons donc proposer une structure précise pour chaque gène cloné, et confirmer ou corriger les prédictions correspondantes s'il y a lieu (figure 3.1.2).

Une fois les gènes clonés dans le vecteur de référence de Gateway, nous sommes en mesure de transférer les inserts dans les vecteurs d'expression de notre choix. Au Dana Farber Cancer Institute, dans le cadre du projet de cartographie des interactions entre protéines de *C. elegans* décrit dans la partie 3.2, nous utilisons en particulier deux vecteurs spécifiques du système double hybride. Mais il est tout à fait envisageable de transférer les gènes clonés vers d'autres vecteurs de destination, dans le cadre d'autres projets de génomique fonctionnelle. Notre projet de clonage de l'ORFéome peut donc aboutir à la constitution d'une ressource profitable pour la communauté *C. elegans*, au-delà de son utilité au sein

du laboratoire.

3.1.c Résultats

Le consortium de séquençage de *C. elegans* (http://www.wormbase.org) prédit actuellement l'existence de 18959 gènes, à l'aide du logiciel GeneFinder. Par ailleurs, 9356 gènes ont été identifiés par le projet de séquençage de cDNAs dirigé par Yuji Kohara à Mishima, Japon [Thi01b], et 784 gènes ont été clonés, séquencés et caractérisés par la communauté *C. elegans*, pour un total de 9503 gènes dont l'existence est expérimentalement confirmée. Parmi ceux-ci, 796 ne sont pas prédits. Les 8707 (9503-796) restants correspondent à 9071 gènes prédits. Ceci illustre le fait que GeneFinder a moins tendance à fusionner deux gènes réels en un gène prédit, que le contraire. Il reste donc 9888 (18959-9071) gènes prédits pour lesquels aucune confirmation expérimentale n'est disponible.

Il est possible que ces gènes prédits soient des faux positifs produits par GeneFinder. Mais aussi, il peut s'agir de vrais gènes faiblement exprimés, non découverts par le projet de séquençage de cDNAs à cause de leur faible taux d'expression. Dans ce cas, notre approche, qui repose sur une amplification par PCR, pourrait tout de même permettre de les détecter et de les cloner. Cette hypothèse est confirmée par nos premiers résultats, décrits ci-dessous, qui concernent un échantillon de près de 3000 gènes.

Tout d'abord, nous avons testé notre logiciel de conception d'amorces et nos conditions de PCR sur un jeu de 988 gènes confirmés, fortement exprimés et pour lesquels la séquence d'un transcrit complet est disponible. La très grande majorité (97%) des amplifications de cet échantillon a produit un fragment de la taille attendue. Ensuite, pour évaluer la qualité de notre banque de cDNAs, nous avons tenté d'amplifier 601 gènes choisis parmi les 784 caractérisés par la communauté *C. elegans*. Comme ces gènes ont généralement été identifiés génétiquement, on peut considérer que cet échantillon possède un large éventail de taux d'expression. Dans la mesure où 91% d'entre eux ont été amplifiés avec succès, on peut considérer que la bonne qualité de notre banque est confirmée.

Nous avons alors appliqué notre approche d'amplification et de clonage à un échantillon aléatoire de 1222 gènes prédits et non confirmés. Parmi ceux-ci,

66% ont été amplifiés avec succès, bien que les tailles des fragments ne correspondent pas toujours à la taille attendue. Dans ces cas, après alignement des traces sur le génome, il apparaît qu'en général les extrémités des gènes sont correctement prédites mais que leurs structures internes sont erronées. De cette manière, nous avons pu proposer des corrections pour 12% des exons séquencés, qui concernent 27% des 564 gènes analysés.

Par ailleurs, en conséquence de ces observations expérimentales, nous pouvons évaluer le nombre de gènes chez *C. elegans*. En effet, l'analyse par OSTs nous conduit à estimer que *C. elegans* possède au moins 17300 gènes [Reb01]. Ce résultat peut paraître surprenant, dans la mesure où la drosophile n'est censée posséder que 13600 gènes et les dernières estimations chez l'Homme sont inférieures à 50000. Cette confirmation du nombre relativement élevé de gènes chez *C. elegans* peut conduire à une remise en cause des évaluations effectuées dans d'autres organismes. De plus, si ces estimations étaient confirmées expérimentalement, il serait très intéressant de les expliquer.

Cette étude préliminaire a suggéré la faisabilité d'un projet de clonage de l'ORFéome complet de *C. elegans*. Un tel projet est actuellement en cours au Dana Farber Cancer Institute, et nous avons récemment franchi la barre des 10000 tentatives de clonage. Remarquons qu'une partie des gènes clonés avec succès par notre approche peuvent tout de même être tronqués par rapport au véritable gène, dans les cas où les exons extrêmes prédits par GeneFinder existent bien mais sont en réalité des exons internes du gène (prédictions incomplètes). Ainsi, cette approche ne devrait pas se substituer aux projets de séquençage systématique de transcriptomes, mais plutôt les complémenter.

3.2 Interactions protéine-protéine

Les interactions protéine-protéine jouent un rôle fondamental dans la majorité des processus biologiques, depuis la formation des structures macromoléculaires et des complexes enzymatiques jusqu'à la régulation des voies métaboliques ou la transmission de signaux biologiques. Pour déterminer la fonction d'une protéine, il est donc essentiel de savoir avec quelles autres protéines celle-ci est susceptible d'interagir. Plusieurs méthodes expérimentales ont été développées afin de tenter de répondre à cette question. Parmi ces méthodes, le système dit "double hybride" [Fie89] s'impose dans les laboratoires comme la technique de choix, en particulier dans le cadre des projets de génomique fonctionnelle. Ce système s'avère en effet applicable à l'échelle d'un génome complet, comme le montrent les études menées sur la levure *Saccharomyces cerevisiae* [Ito00], [Uet00]. Pour mener à bien un tel projet de génomique fonctionnelle, il est néanmoins nécessaire de développer une plate-forme informatique pour générer, stocker et présenter les résultats. Dans cette partie, nous rappelons tout d'abord le principe du système double hybride et les possibilités de son application à l'échelle d'un génome. Nous présentons ensuite le projet, en cours au Dana Farber Cancer Institute, de cartographie des interactions entre protéines chez *C. elegans*, en particulier dans le cadre des protéines impliquées dans le développement de la vulve [Wal00]. Nous détaillons enfin la plate-forme informatique mise en place pour ce projet.

3.2.a Le système double hybride

La technique double hybride permet d'étudier directement, grâce à la génétique de la levure, les interactions entre protéines provenant d'organismes quelconques. L'idée repose sur l'utilisation d'un facteur de transcription issu de la levure *Saccharomyces cerevisiae* : la protéine Gal4p. Dans une levure sauvage en présence d'un sucre, le galactose, cette protéine est produite et active la transcription du gène Gal1, responsable de la digestion du galactose. Comme

beaucoup d'autres facteurs de transcription, Gal4p comprend deux domaines fonctionnels physiquement distincts et séparables : un domaine de liaison à l'ADN, notée DB (pour dna-binding), et un domaine d'activation de la transcription (AD pour activation domain). DB se fixe sur une séquence spécifique d'ADN, la séquence activatrice en amont du promoteur de Gal1 (GUAS, Gal1 upstream activating sequence). Il a été démontré que dans la levure, la partie intermédiaire de la protéine Gal4p ne modifie pas son action : toute protéine comportant DB en extrémité et AD vers l'autre extrémité active Gal1. En fait, il n'est même pas nécessaire que la liaison entre DB et AD soit covalente. Gal1, ainsi que tout gène comportant la séquence GUAS en amont de son promoteur, sera activé par la présence de DB et AD maintenus à proximité l'un de l'autre. Ainsi, si X et Y sont deux protéines qui interagissent, il est possible de reconstituer un facteur de transcription fonctionnel équivalent a Gal4p en exprimant dans la cellule les deux protéines hybrides DB-X et Y-AD (d'où le nom du système).

Pour déterminer si une protéine X interagit avec une protéine Y, l'idée est donc de transformer des levures avec deux plasmides. L'un est capable de produire la protéine de fusion DB-X, et l'autre la protéine Y-AD. Le génome de la levure est préalablement modifié par ingénierie génétique pour que celle-ci ne puisse plus produire Gal4p. Dans ces conditions, la levure ne pourra dégrader le galactose que si X et Y interagissent. Cependant la capacité de dégradation du galactose est difficile à observer, car la levure dispose de plusieurs mécanismes métaboliques.

Heureusement, nous disposons chez *S. cerevisiae* de plusieurs marqueurs sélectifs. Il s'agit de gènes dont l'expression peut être très facilement observée, car dans certaines conditions elle est nécessaire a la survie de la cellule. Par exemple, le gène Ura3 code pour une enzyme nécessaire à la synthèse de l'uracile. Sur un milieu ne contenant pas d'uracile, les cellules incapables de produire la protéine Ura3p ne peuvent donc pas se développer. En plaçant un ou plusieurs gènes rapporteurs sélectifs sous le contrôle de GUAS par ingénierie génétique, il devient alors facile d'observer l'interaction entre DB-X et Y-AD.

A l'aide du système double hybride, il est donc possible de constater l'interaction entre deux protéines. A l'échelle d'un génome, cette technique peut s'appliquer de deux manières différentes.

La première approche, dite matricielle, consiste à cloner un nombre relativement élevé de gènes dans les deux vecteurs utilisés en phase avec DB ou AD. Il suffit alors de tester chacun des gènes contre tous les autres. Concrètement, pour chaque paire DB-X/Y-AD, une colonie diploïde de levure est générée par croisement et ses phénotypes double hybride sont évalués (résistance sur un milieu URA- par exemple). Comme les colonies positives correspondent à des paires X/Y connues, les interacteurs sont identifiés directement.

La seconde approche repose sur un criblage génétique classique. Elle nécessite la construction préalable d'une banque de cDNAs, clonés dans un vecteur en fusion avec AD. Chaque protéine étudiée, appelée appât, est clonée en fusion avec DB. Des cellules de levure sont transformées avec une copie du plasmide DB et un des plasmides AD au hasard. Les cellules sont cultivées sur un milieu sélectif correspondant à l'un des gènes rapporteurs utilisés, puis les phénotypes double hybride sont évalués pour les colonies survivantes. Dans cette approche, les inserts des plasmides AD positifs doivent alors être séquencés afin d'identifier les partenaires de notre appât.

Les deux approches présentent des avantages et des inconvénients. L'approche matricielle est initialement plus lourde, puisqu'elle nécessite d'avoir cloné un grand nombre de gènes avant toute expérience. Par contre, elle est bien plus efficace à moyen terme, une fois l'étape de clonage effectuée, car elle évite la pénible et coûteuse phase de séquençage et d'identification des interacteurs. Cependant l'approche matricielle ne permet de travailler qu'avec des gènes identifiés, et se trouve donc limitée par la qualité des prédictions de zones codantes dans l'organisme d’intérêt. Au contraire, l'approche par criblage n'est limitée que par la qualité de la banque de cDNAs utilisée. Même, elle produit au passage un équivalent des ESTs, et peut entraîner la découverte de gènes. Par ailleurs, l'approche par criblage repose sur l'utilisation d'une banque de cDNAs. Les interacteurs obtenus correspondent donc souvent à des protéines partielles. D'un coté, ceci peut augmenter la sensibilité du système, en révélant des interactions qui nécessitent dans la protéine complète une modification post-traductionnelle afin d'exposer ou d'activer le domaine d'interaction. D'un autre coté, cette approche peut procurer un certain nombre de faux positifs, correspondant en fait à des peptides courts sans rapport avec le protéome étudié. En effet, rappelons qu'après clonage d'une banque de cDNAs de façon non

directionnelle dans un vecteur d'expression, 5 plasmides sur 6 contiennent un insert dans le mauvais sens ou hors-phase. L'approche par criblage nécessite donc une analyse minutieuse des traces de séquençage correspondant aux interacteurs potentiels.

3.2.b Cartographie des interactions entre protéines chez *C. elegans*

Dans le cadre du projet de cartographie des interactions protéine-protéine chez *C. elegans*, nous employons une version du système double hybride optimisée pour une utilisation à grande échelle et haut débit. Le principal souci a été de limiter au maximum la production de faux positifs qui, s'ils peuvent gêner le biologiste traditionnel, constituent un véritable handicap pour un projet de génomique fonctionnelle. Nous utilisons les trois gènes rapporteurs Ura3 et His3 de la levure, et LacZ qui provient d'*E. coli*. Ura3 et His3 sont des marqueurs sélectifs, et LacZ code pour la beta-galactosidase, une enzyme capable de convertir le substrat incolore X-Gal en un colorant bleu. L'utilisation de trois gènes, tous sous contrôle de GUAS mais comportant par ailleurs des contextes de promoteurs très différents, permet d'augmenter considérablement la spécificité de l'expérience de double hybride. De plus, nous utilisons des vecteurs centromériques, dont le nombre de copies reste faible dans la levure. Nous limitons ainsi l'ampleur des problèmes posés par la sur-expression des protéines de fusion DB-X et Y-AD.

Dans un premier temps, nous nous sommes intéressés à 29 protéines impliquées dans le développement de la vulve chez *C. elegans* [Wal00]. Au moins quatre voies biologiques fonctionnent en coordination pour arriver à la formation d'une unique vulve chez l'adulte hermaphrodite : une voie RTK/Ras, une voie Notch, et deux voies fonctionnellement redondantes multivulve synthétiques, synMuv A et synMuv B. Ce processus biologique a été largement étudié par la communauté *C. elegans*. Plusieurs interactions entre protéines y sont identifiées et jouent un rôle crucial. Cette connaissance peut permettre une évaluation convenable du taux de faux-négatifs générés par la méthode. Toutefois de nombreuses protéines impliquées dans le processus restent peu ou pas

caractérisées. Dans ce cadre notre étude peut bien entendu s'avérer très utile. Enfin, les homologues de certaines des protéines impliquées dans le développement de la vulve chez *C. elegans* jouent un rôle dans l'oncogénèse chez l'Homme, notamment LIN-35 (protéine du rétinoblastome pRB). Dans cette mesure, l'étude de ce processus biologique pourrait donc améliorer la compréhension du développement de certains cancers chez l'Homme. L'étude de ce processus constitue donc un très bon contexte pour un projet pilote de cartographie des interactions entre protéines chez *C. elegans*.

Les deux approches de double hybride décrites dans la partie 3.2.a ont été appliquées. D'une part, les 29 protéines d'intérêt ont été clonées dans les deux vecteurs, en fusion avec DB et AD, à l'aide du système Gateway (voir partie 1). La matrice 29x29 a détecté plus de 50% (6 sur 11) des interactions décrites dans la littérature. Les cas non détectés peuvent généralement s'expliquer par les propriétés du système double hybride ou la physiologie des cellules de levure. Deux nouvelles interactions intéressantes ont été détectées, LIN-10/LIN-10 et LIN-53/LIN-37. D'autre part, 27 des 29 protéines ont servi d'appât pour l'approche par criblage (deux protéines activent la transcription d'elles-mêmes et ont donc été écartées). 148 interactions potentielles ont été détectées, mettant en cause 124 interacteurs différents, dont 15 étaient précédemment caractérisées génétiquement. Cette approche permet donc de proposer une annotation fonctionnelle pour les 109 autres interacteurs.

3.2.c La base de données WISTdb

Dans le cadre du projet de cartographie des interactions entre protéines chez *C. elegans*, nous sommes amenés à manipuler de grandes quantités de données. En particulier, la méthode double hybride par criblage produit des centaines de séquences d'ADN correspondant à des dizaines d'appâts. Ces séquences doivent être alignées sur le génome de *C. elegans* afin d'identifier les interacteurs potentiels. Les traces de séquençage associées doivent être examinées afin de vérifier que les inserts sont dans le bon cadre de lecture. L'ensemble des paramètres expérimentaux et des résultats doit être conservé de manière durable, fiable et standardisée, pour limiter les erreurs humaines liées à la quantité des

données traitées et permettre la comparaison et la fusion des résultats provenant de chercheurs différents (travail coopératif). Enfin, comme pour tout projet de génomique fonctionnelle, il est essentiel que la communauté scientifique ait accès aux données produites dans notre laboratoire, dans la mesure où l'analyse des résultats ne peut se faire localement mais doit plutôt être effectuée de manière répartie dans les laboratoires de génétique intéressés par tel ou tel gène.

Pour répondre à tous ces besoins, nous avons développé une plate-forme informatique basée sur le système de gestion de base de données orienté-objet Acedb (voir 3.1.b). Nous utilisons également le système Acembly [Acembly] d'assemblage et d'édition de séquences, pour l'édition des traces de séquençage et leur alignement sur le génome de *C. elegans*. Rappelons qu'Acembly est une extension d'Acedb développée pour analyser les résultats issus du projet de séquençage systématique de cDNAs chez *C. elegans* du professeur Y. Kohara [Thi01b]. Le protocole utilisé dans notre laboratoire est le suivant.

Pour chaque série de criblages double hybride, une base de données locale, appelée WISTdb (pour Worm Interaction Sequence Tag database), est construite. Elle contient un sous-ensemble des données publiques concernant *C. elegans*, notamment la séquence génomique et les informations sur les gènes prédits ou caractérisés. L'ensemble des séquences correspondant aux plasmides positifs du criblage est aligné sur le génome automatiquement. Les informations concernant les conditions expérimentales, les phénotypes observés et les appâts utilisés sont saisis via une interface conviviale ou générés automatiquement par le système. Pour chaque ensemble de clones provenant du même appât et s'alignant sur le même gène, un clone représentatif, généralement le clone le plus long, est sélectionné. L'utilisateur doit alors vérifier le cadre de lecture des clones représentatifs à l'aide d'Acembly (voir figure 3.2.1), puis retester les phénotypes double hybride de ces clones avec leurs appâts respectifs. Enfin, les interactions potentielles ainsi confirmées sont annotées, et constituent les ISTs (Interaction Sequence Tags). Dans la base WISTdb courante, de nouveaux objets sont automatiquement créés pour représenter ces ISTs. Les ISTs provenant du même appât sont regroupés dans des objets ISTScreen. Finalement, l'ensemble de la base courante est exportée vers la base WISTdb principale, qui regroupe l'ensemble des résultats produits au laboratoire.

Pour mettre les résultats obtenus à la disposition de la communauté

Figure 3.2.1 : Vérification du cadre de lecture d'un clone obtenu par double hybride avec l'appât lin-53.

A gauche, la structure complète exons (boites) / introns (traits) est représentée, ainsi que la séquence génomique (petites lettres au centre) et la trace de séquençage pour la zone examinée. Il s'agit de l'extrémité 5' du clone (les séquences sont représentées 5' en haut et 3' en bas). La séquence du vecteur a été identifiée et la trace correspondante est masquée. Le vecteur se termine par le codon TCC, puis un G seul. La protéine exprimée l'est donc dans le cadre de lecture des codons (G)CA GAA.

A droite, une vue d'ensemble représente, de gauche à droite : les traces de séquençage alignées à cet endroit dans le génome (traits verticaux, avec un intron identifié), les traductions hypothétiques dans les trois cadres de lecture possibles, et la séquence génomique. On peut constater que le codon GAA en question correspond à l'acide aminé E du troisième cadre de lecture représenté. Il s'agit bien d'un cadre ouvert de lecture (les deux autres comportent rapidement des * représentant des codons stop). Ce clone est donc en phase.

scientifique, nous tirons parti des systèmes AcePerl et AceBrowser [Ste99].

AcePerl est une interface à Acedb pour la programmation en langage Perl. AceBrowser utilise AcePerl et propose une interface web hautement configurable à toute base de données Acedb. A l'aide de ces systèmes, nous avons conçu et mis en œuvre une interface web adaptée spécifiquement au format de données utilisé dans WISTdb (voir http://vidal.dfci.harvard.edu).

Etant donné qu'une partie des résultats présents dans la base WISTdb principale doit rester temporairement confidentielle jusqu'à publication, l'interface permet en fait d'accéder à la base WISTdb publique, qui est une copie partielle de la base principale. Des programmes ont été développés pour exporter, importer et fusionner facilement tout ou partie des diverses bases WISTdb. La base publique actuelle, disponible à l'adresse http://vidal.dfci.harvard.edu, comporte exclusivement les résultats des expériences sur les protéines impliquées dans le développement de la vulve chez *C. elegans*, mais des mises à jour régulières sont prévues.

Il est à noter que le système informatique complet est facilement réutilisable. Une collaboration récente avec Anne Davy [Dav01], du CRBM de Montpellier, qui a étudié les interactions entre protéines du protéasome de *C. elegans*, l'a bien montré.

3.3 Conclusion

Dans ce chapitre, deux projets de génomique fonctionnelle ont été décrits, sur lesquels nous avons travaillé dans le laboratoire du docteur Marc Vidal, au Dana Farber Cancer Institute, Boston.

D'une part, le projet de clonage de l'ORFéome de *C. elegans* a été présenté. Ce projet repose sur une amplification par PCR de zones codantes prédites, suivie du clonage des produits obtenus dans le vecteur navette du système Gateway de clonage par recombinaison. A cette fin, nous avons développé un programme informatique pour automatiser l'étape de conception des amorces de PCR. Dans le cadre du système Gateway, il est possible de transférer les gènes clonés vers un grand nombre de vecteurs d'expression. Dans notre laboratoire, nous utilisons en particulier comme vecteurs de destination deux vecteurs spécifiques du système double hybride, dans le cadre du projet de cartographie des interactions entre protéines. Mais plus généralement, notre projet de clonage de l'ORFéome doit aboutir à la constitution d'une ressource qui pourrait s'avérer très utile pour d'autres projets de génomique fonctionnelle chez *C. elegans*.

D'autre part, le projet de cartographie des interactions entre protéines de *C. elegans* a été décrit, à travers un projet pilote concernant des protéines impliquées dans le développement de la vulve. Le système double hybride apparaît comme une bonne solution pour l'identification des interactions entre protéines dans un cadre de génomique fonctionnelle. Le projet pilote de cartographie des interactions entre protéines impliquées dans le développement de la vulve chez *C. elegans* nous a conduit à développer des protocoles et une plate-forme informatique, nommée WISTdb, susceptibles de s'appliquer à l'échelle d'un génome.

Forts de cette expérience, nous travaillons actuellement sur plusieurs cartes d'interaction de taille moyenne dans le laboratoire du professeur Marc Vidal, au Dana Farber Cancer Institute. Parallèlement, le projet de clonage de l'ORFéome décrit dans la partie 3.1 suit son cours, et permet d'envisager à moyen terme une cartographie complète des interactions chez *C. elegans*, en utilisant

l'approche matricielle du système double hybride.

4. InterDB : une base de données pour la prédiction d'interactions protéine-protéine

Nous avons décrit dans le chapitre 3 le projet de cartographie des interactions entre protéines chez *C. elegans*, en cours au Dana Farber Cancer Institute. Dans le cadre de ce projet, je participe à la détermination expérimentale à grande échelle d'interactions protéine-protéine. A ce jour, plusieurs centaines d'interactions ont été découvertes. Cependant, pour étudier globalement les interactions entre protéines, et au final être capable d'aider la recherche expérimentale en fournissant des prédictions d'interactions, il est nécessaire de disposer du plus grand nombre possible d'exemples, en provenance d'organismes variés, pour pouvoir exploiter la conservation observée des séquences des protéines au cours de l'évolution, ainsi que la conservation soupçonnée de leurs fonctions et de leurs interactions. Dans cette optique, j'ai construit la base de données multi-organismes d'interactions protéine-protéine InterDB, fédérant les données issues de notre projet chez *C. elegans* et d'autres données expérimentales provenant de bases de données publiques. Dans ce chapitre, nous exposons les objectifs, la méthode employée et la mise en œuvre, ainsi que les résultats relatifs à la construction de cette base de données.

4.1 Objectifs et approches

Pour étudier les interactions entre protéines de manière globale, il est nécessaire de disposer d'une base de données comprenant le plus grand nombre possible d'exemples de couples de protéines qui interagissent effectivement. Nous disposons naturellement des interactions entre protéines de *C. elegans* découvertes dans le cadre du projet mené par le docteur Vidal, mais il est souhaitable d'incorporer des exemples d'interactions expérimentalement vérifiées provenant d'autres sources. Il s'agit donc de construire une base fédérant des données biologiques d'origines diverses.

De plus, dans la mesure où nous souhaitons utiliser cette base de données comme support d'une analyse prédictive des interactions entre protéines, il est nécessaire d'y incorporer des informations descriptives des protéines susceptibles d'expliquer les interactions observées.

Pour répondre aux objectifs fixés, les problèmes à résoudre sont les suivants. Il s'agit d'abord d'obtenir des données d'interaction protéine-protéine. Il est ensuite nécessaire d'identifier les protéines sans ambiguïté. Il est enfin essentiel de caractériser ces protéines.

4.1.a Interactions entre protéines

Il est difficile de trouver des données d'interactions entre protéines qui soient utilisables dans un contexte informatique, car les biologistes étudient généralement la fonction d'une protéine d'intérêt. Dans ce cadre, ils sont souvent amenés à rechercher et identifier des partenaires d'interaction de cette protéine. A terme, cette information est publiée, mais elle est rarement mise en avant dans les articles scientifiques. Il est évident qu'une étude manuelle systématique de la littérature dans le but de rechercher ce type de résultat n'est pas une option pour un informaticien travaillant seul. On peut envisager de mener une analyse automatique de la littérature, en appliquant des techniques d'extraction d'informations à partir de textes en langage naturel. C'est l'approche adoptée

notamment par Marcotte *et al* [Mar01] pour extraire des données d'interaction protéine-protéine à partir de résumés de la base publique de références PubMed. Malheureusement, bien que cette approche soit prometteuse, elle ne peut fournir que des *suspicions* d'interactions, de par l'étape informatique de traitement des textes en langage naturel. Or, il est essentiel que InterDB ne contienne que des interactions sûres, c'est-à-dire déterminées expérimentalement.

Pour obtenir de telles données, il est nécessaire soit de s'associer à des projets de génomique fonctionnelle qui en produisent, ce que nous avons fait avec l'équipe du docteur Vidal au DFCI, soit de trouver des bases de données compilées par des tiers, en général par curation manuelle de la littérature scientifique. Une recherche dans la littérature et sur internet nous a conduits à utiliser les bases de données suivantes.

La base DIP (Database of Interacting Proteins, [Mar99], [Xen01]), créée et maintenue par le groupe de David Eisenberg à UCLA, contient un ensemble d'interactions expérimentalement établies entre protéines de divers organismes. Les interactions proviennent généralement de la littérature, et les références bibliographiques sont fournies. Jusqu'en septembre 1999, les protéines étaient référencées uniquement par leurs identificateurs PIR (Protein Information Resource, [Win99]). Les dernières versions de DIP utilisent également parfois les identificateurs SwissProt. La base complète est disponible gratuitement sur internet, et contient environ 2290 interactions à ce jour.

Une autre source d'interactions utilisée est la base YPD (Yeast Protein Database, [YPD] [Hod99]). Il s'agit d'une base de données générale sur les protéines de *S. cerevisiae* résultant d'une analyse manuelle de la littérature. YPD est une base privée, gratuitement accessible aux utilisateurs académiques. Elle contient une grande quantité d'annotations fonctionnelles diverses, dont en particulier des interactions protéine-protéine. Malheureusement ces informations sont en langue naturelle (en anglais), et tout accès scripté à YPD est interdit, ce qui exclut a priori son utilisation dans notre contexte. Toutefois, un accord a été négocié avec Proteome Inc. nous donnant accès à un tableau récapitulatif de toutes les interactions présentes dans la base. Ce tableau comprend 1115 interactions, et les protéines partenaires y sont référencées par les noms des gènes.

Enfin, certaines bases d'interactions entre protéines ont été identifiées mais n'ont pas été utilisées. C'est notamment le cas de la base FlyNets [San99], qui

contient à ce jour 53 interactions protéine-protéine chez *D. melanogaster*.

4.1.b Identification des protéines

Le second problème rencontré est une conséquence du constat suivant : les bases de données biologiques sont hautement redondantes, sans que cela soit explicite. En effet, de nombreuses bases décrivent les mêmes objets biologiques en utilisant des identificateurs différents. Ce problème est particulièrement gênant dans la mesure où nous cherchons à créer une base fédérant des données provenant de diverses sources.

Par exemple, une protéine peut posséder plusieurs noms, ainsi que des numéros d'identification différents dans SwissProt, GenBank et PIR. Il n'est généralement pas trivial d'établir le lien entre ces objets a priori différents. Ainsi, deux interactions provenant par exemple de DIP et de YPD peuvent mettre en cause une protéine commune à notre insu.

Pour résoudre ce problème et assurer la cohérence d'InterDB, nous avons choisi d'utiliser les bases de données complémentaires SwissProt et TrEMBL [Bai99] comme squelette d'InterDB. TrEMBL est une base de protéines putatives construite par traduction et annotation automatique des séquences nucléiques de l'EMBL/EBI. Après curation manuelle et annotation plus fine par des experts humains, les protéines de TrEMBL sont transférées dans SwissProt. Les interactions entre protéines provenant des bases source (WISTdb, YPD, DIP) ne sont conservées dans InterDB que si les deux partenaires qui interagissent sont tous les deux identifiées dans SwissProt ou TrEMBL. Cette stratégie conduit à écarter éventuellement un certain nombre d'interactions par ailleurs intéressantes, mais nous garantit la cohérence d'InterDB.

4.1.c Caractérisation des protéines

Rappelons que InterDB doit servir à une étude prédictive des interactions entre protéines. Dans cette optique, elle doit contenir des informations descriptives des protéines, susceptibles d'expliquer les interactions observées et

donc d'en prédire d'autres. Dans la version actuelle d'InterDB, nous avons choisi d'utiliser comme descripteurs les domaines et familles de protéines de la base Interpro, ainsi que la localisation cellulaire et les mots-clés issus de SwissProt.

Les domaines et familles de protéines décrits dans des bases comme Pfam [Bat99], ProSite [Hof99], Prints [Att97] [Att99], Blocks [Hen98] [Hen99], ou encore ProDom [Cor99] nous paraissent revêtir une grande importance pour les interactions entre protéines. En effet, ces domaines ont généralement été caractérisés initialement en biologie classique, et ont donc indubitablement une signification biologique. Qui plus est, certains domaines ont été identifiés pour leur rôle dans les interactions entre protéines. C'est notamment le cas du domaine SH2, qui est connu pour interagir avec des peptides contenant des phosphotyrosines.

Partant de cette conviction, nous avons choisi d'intégrer une partie de ces bases dans InterDB. Il est intéressant de remarquer que si ces bases ont toutes plus ou moins le même objectif, elles reposent sur des modélisations diverses et complémentaires, allant des modèles de Markov cachés (voir par exemple [Dur98]) pour Pfam jusqu'au clustering automatique de séquences protéiques par PSY-BLASTs récursifs chez ProDom, en passant par les expressions régulières et les matrices poids-position de ProSite. Les résultats obtenus sont donc parfois redondants, souvent complémentaires. Dans les versions initiales d'InterDB, nous nous sommes restreints aux bases Pfam et ProSite.

Récemment, les responsables de certaines bases de domaines et familles protéiques se sont associés pour produire la base Interpro [Apw01], qui fédère les données de Pfam, ProSite, Prints et ProDom. Nous avons surveillé cet effort avec un grand intérêt et avons intégré les données d'Interpro dans InterDB, en remplacement de Pfam et ProSite, dès la version 2.0 d'Interpro.

Par ailleurs, les questions de cohérence introduites en 4.1.b.2 nous ont conduits à utiliser SwissProt et TrEMBL comme squelette d'InterDB. En plus de fournir un référentiel stable pour les protéines, ces bases comportent des annotations que nous avons choisi d'exploiter.

En particulier, les protéines y sont caractérisées par des mots-clés, choisis dans un vocabulaire comportant environ 840 termes. Ces mots-clés sont extrêmement hétérogènes, en termes de spécificité et de sémantique. Ainsi, le mot-clé *Hypothetical_protein* signifie simplement qu'il n'existe aucune

confirmation expérimentale de l'existence de la protéine en question. Ce mot-clé est présent dans plus de 15000 protéines, et n'a clairement aucun rapport avec les interactions entre protéines. Par contraste, le mot-clé *SH3-binding* décrit 20 protéines, et présente pour nous un intérêt évident. Les mots-clés SwissProt peuvent donc contenir des informations intéressantes, et bien qu'ils doivent être interprétés avec précaution, nous avons choisi de les incorporer comme descripteurs dans InterDB. Une liste exhaustive des mots-clés SwissProt, ainsi que leurs significations, sont disponibles sur le site web de SwissProt.

De même, la localisation cellulaire des protéines est parfois annotée dans le champ "commentaires" de SwissProt et TrEMBL. Cette information présente un grand intérêt, dans la mesure où deux protéines sont d'autant plus susceptibles d'interagir qu'elles sont co-localisées dans la cellule. Cette information est fournie en langage naturel, mais une analyse informatique simple permet d'extraire la localisation dans la grande majorité des cas (voir 4.2.b.1). Nous disposons ainsi de 100 descripteurs supplémentaires pour les protéines d'InterDB.

4.2 Mise en œuvre

Nous disposons de diverses sources de données, qu'il faut stocker dans InterDB. Il s'agit donc d'une part de définir un schéma pour ces données, et d'autre part de concevoir et de développer des programmes pour extraire des bases d'origine l'information concernant les protéines et les interactions.

4.2.a Conception du schéma d'InterDB

Pour gérer InterDB, nous avons choisi d'utiliser le système de gestion de bases de données acedb [Acedb], [Dur94], [Thi99]. Ce choix résulte principalement de notre expérience convaincante lors de la construction de WISTdb (voir partie 3.2.c). De plus, acedb est particulièrement bien adapté pour gérer des données en constante évolution : il est facile de modifier un schéma acedb en rajoutant des classes ou des attributs, sans que cela nécessite une recompilation des données. Enfin, acedb offre un format d'échange simple et puissant pour l'importation et l'exportation de données, le format *.ace*.

Dans les grandes lignes, les classes principales sont ?Protein, qui permet de représenter les protéines ainsi que leurs descriptions, et ?Interaction qui permet de représenter les interactions entre protéines et comprend principalement les deux attributs Protein1 et Protein2, pointant vers des objets de la classe ?Protein. Ce qui nous a guidé sont les deux considérations suivantes.

Dans le cadre de la conception du schéma d'InterDB, le premier objectif est la représentation de l'ensemble des connaissances souhaitées. Nous voulons en particulier stocker les informations extraites de SwissProt et TrEMBL susceptibles de faciliter l'identification des protéines, comme par exemple les noms de gènes et les références croisées vers d'autres bases protéiques comme PIR. Nous souhaitons aussi conserver des informations qui caractérisent les protéines : les descripteurs bien sûr (voir 4.1.b.3), mais aussi d'autres données qui pourraient s'avérer utiles, à savoir la séquence peptidique primaire, l'organisme d'origine et les protéines homologues putatives.

Par ailleurs, il est clair que le modèle choisi joue un rôle crucial en ce qui concerne le temps d'exécution des requêtes. Or, dans notre cas, InterDB doit servir à prédire des interactions entre protéines. Dans ce cadre, nous sommes amenés à effectuer quelques requêtes typiques un très grand nombre de fois. Il est important de concevoir un schéma qui prenne en compte cet aspect, et qui permette d'optimiser ces requêtes. Par exemple, les noms de gènes sont stockés dans la classe ?GeneName, qui contient simplement un attribut Protein pointant vers la ou les protéines portant le nom en question. Cette modélisation permet d'accéder très rapidement à la protéine codée par le gène de nom XYZ, alors que cette requête nécessiterait un parcours des 400 000 protéines dans le cas du modèle naïf où GeneName serait un attribut textuel de la classe ?Protein.

Mises à part ces considérations, le schéma ne présente pas de difficultés de conception, et ne mérite pas d'explications plus détaillées. Il est reproduit dans son intégralité et assorti de commentaires en annexe (annexe 4.1).

4.2.b Acquisition des données d'InterDB

InterDB est une base fédérative construite à partir de bases source qui évoluent régulièrement, quantitativement mais aussi qualitativement. Il est donc essentiel de disposer d'un système logiciel de construction d'InterDB qui soit efficace, pour autoriser des mises à jour fréquentes, et souple, pour permettre d'incorporer de nouveaux types de données facilement.

Pour chaque base source : SwissProt, TrEMBL, Interpro, WISTdb, YPD, et DIP, nous avons développé un interpréteur, qui effectue une analyse syntaxique et sémantique des fichiers source, et génère les objets correspondants au format *.ace* conformément au schéma de InterDB. Ces programmes peuvent être classés dans deux familles, selon qu'il s'agisse de données sur les protéines ou de données d'interaction.

4.2.b.1 Acquisition des données protéiques

Dans le cas des données protéiques, l'objectif est d'analyser les fichiers source, et d'en extraire les connaissances qui caractérisent les protéines et qui sont

susceptibles d'expliquer les interactions observées. Nous exploitons le fait que les bases source utilisées, à savoir SwissProt, TrEMBL et Interpro, sont semi-structurées : une étude des documentations et des fichiers source permet de mettre cette structure en évidence.

La structure est souvent rigide : c'est le cas pour 10 des 11 champs de SwissProt et TrEMBL analysés. Dans ce cas une analyse syntaxique basée sur les expressions régulières est suffisante. Sa mise en œuvre est extrêmement facilitée par le langage Perl. Par exemple, la documentation utilisateur de SwissProt indique que les numéros d'accession des protéines sont stockés dans une ligne de la forme :

AC <AC1> [; <AC>]*.

où <AC1> est le numéro d'accession primaire, et les éventuels numéros suivants, séparés par des points-virgules, sont les numéros d'accession secondaires. L'acquisition des numéros d'accession est facilement accomplie à l'aide d'une expression régulière Perl, directement inspirée de cette définition provenant de la documentation utilisateur de SwissProt.

Quant à la base Interpro, elle s'avère fortement structurée puisqu'elle est distribuée au format XML. Une analyse à l'aide de 18 expressions régulières permet d'en extraire toute l'information utile : descriptions textuelles des domaines et familles, relations horizontales et verticales entre eux, et protéines de SwissProt et TrEMBL qui les contiennent.

Par contre, si la structure est molle, par exemple dans le cas des commentaires en texte libre de SwissProt, l'extraction des connaissances peut a priori sembler nécessiter l'application de techniques d'analyse de textes en langage naturelle. Nous souhaitons éviter cette solution, principalement car le traitement automatique de textes en langage naturel constitue un domaine de recherches en soi, qui n'est pas le nôtre.

Un cas particulier est celui de la localisation cellulaire des protéines, qui est fournie en texte libre dans le champ CC de SwissProt et TrEMBL. Nous considérons que cette information est importante. Par exemple, deux protéines notoirement localisées dans le reticulum endoplasmique sont plus susceptibles d'interagir que deux protéines respectivement membranaire et nucléaire. Nous avons donc cherché à extraire cette information, de manière éventuellement partielle mais exacte. L'approche mise en œuvre est la suivante.

Une analyse du fichier SwissProt permet de constater que ces commentaires de localisation sont en pratique hautement redondants : si l'on se limite par exemple aux localisations qui décrivent au minimum 50 protéines, on obtient moins de 100 descriptions qui couvrent la très grande majorité des protéines annotées. On peut donc remplacer ces informations textuelles inexploitables automatiquement par un vocabulaire contrôlé comportant moins de 100 termes, sans perte d'information significative.

La traduction des fichiers source de SwissProt et TrEMBL est en conséquence effectuée en deux passes : dans une première passe, un dictionnaire des localisations fréquentes est construit ; dans une seconde passe, l'analyseur syntaxique extrait l'information fortement structurée des fichiers source à l'aide d'expressions régulières et, si une localisation est fournie, vérifie que celle-ci est présente dans le dictionnaire avant de l'inclure dans le fichier *.ace* de sortie.

L'approche utilisée pour extraire l'information de localisation est également appliquée aux commentaires libres d'annotation fonctionnelle de SwissProt et TrEMBL, mais il apparaît que ces commentaires sont sémantiquement beaucoup plus hétérogènes. Une approche informatique plus sophistiquée serait nécessaire pour extraire cette information.

4.2.b.2 Acquisition des données d'interaction

Dans le cas des données d'interaction, il est nécessaire d'interroger la base InterDB en cours de construction pour identifier les protéines qui interagissent. Nous utilisons pour cela AcePerl [Ste99], une interface programmeur aux bases acedb en Perl. Ce système est facile à installer et à utiliser, et bénéficie d'une documentation très claire.

Dans les grandes lignes, les interpréteurs des données d'interaction fonctionnent selon l'algorithme suivant :

1. analyse syntaxique du fichier source par expressions régulières, extraction des identificateurs externes (i.e. noms des gènes, numéros d'identification de PIR, noms ID de SwissProt) des protéines qui interagissent;
2. recherche dans InterDB des identificateurs internes (numéros d'accession primaires de SwissProt ou TrEMBL) de ces interacteurs, par génération de requêtes acedb appropriées et utilisation de AcePerl;

3. génération des objets représentant les interactions dans InterDB, par production d'un fichier au format *.ace*.

L'ensemble des programmes développés pour la construction d'InterDB est disponible en annexe 4.2. Ces programmes sont linéaires en temps en fonction de la taille du fichier d'entrée. Les optimisations de temps d'exécution proviennent généralement d'une modification du schéma d'InterDB, comme décrit en partie 4.2.a.

4.3 Résultats

Les résultats de nos travaux pour la construction d'InterDB peuvent être vus sous deux perspectives : d'un point de vue statique, nous disposons aujourd'hui d'une base de données d'interactions entre protéines, dont nous présentons le contenu; à un niveau plus dynamique, nous avons développé un système logiciel performant et réutilisable pour la construction de bases d'interaction protéiques, qui permet des mises à jour fréquentes de InterDB.

4.3.a Contenu de InterDB

La version actuelle de InterDB contient 463 148 protéines, et 2464 interactions concernant 2032 protéines. Remarquons que bien que seulement 2032 protéines sont impliquées dans des interactions, il est nécessaire que InterDB contienne l'ensemble des protéines de SwissProt et TrEMBL afin de permettre l'identification des interacteurs. De plus, InterDB doit servir à la prédiction de nouvelles interactions. Dans ce cadre, nous avons besoin de caractériser toutes les protéines, afin de pouvoir proposer des interacteurs potentiels de protéines d'intérêt (voir partie 5).

Rappelons que les interactions proviennent des trois sources DIP, YPD et WISTdb. Nous pouvons quantifier les apports de ces sources comme suit : 54% des interactions proviennent de DIP, 36% de YPD et 10% de WISTdb. Ces chiffres confirment l'intérêt de notre choix de ne pas nous limiter aux données d'interactions produites au DFCI.

Rappelons aussi que la stratégie choisie pour assurer la cohérence d'InterDB nous conduit à éliminer un certain nombre d'interactions provenant de chaque source (voir 4.2.b.2). En effet, les interactions figurant dans InterDB sont celles pour lesquelles les deux partenaires qui interagissent ont été identifiés dans SwissProt ou TrEMBL. Dans le cas de DIP, où les protéines sont désignées par leur identification SwissProt ou par un numéro PIR, nous devons écarter 42% des interactions. Par opposition, nous ne perdons que 22% et 25% des interactions en

provenance de YPD et WISTdb, respectivement, qui utilisent les noms des gènes pour désigner les protéines. Nous pouvons déduire de ces observations que les références croisées de SwissProt vers PIR sont très mauvaises, alors que les noms de gènes sont relativement bien annotés dans SwissProt.

Nous avons conçu InterDB comme une base multi-organismes d'interactions entre protéines : nous y avons intégré toutes les données d'interaction disponibles, quel que soit l'organisme d'origine des interacteurs. En pratique, nous observons que 70.5% des protéines impliquées dans des interactions proviennent de *Saccharomyces cerevisiae*, 9.9% de l'Homme, 9.3% de *Caenorhabditis elegans*, et les 10.3% restantes proviennent de 41 organismes divers. Nous constatons donc que InterDB contient principalement des interactions chez la levure. Ceci peut s'expliquer en partie par l'utilisation de YPD comme source, mais les nombres observées indiquent que dans DIP aussi, la majorité des interactions proviennent de la levure *S. cerevisiae*, eucaryote unicellulaire. Cette observation n'est pas gênante, mais il faut tout de même en être conscient. En effet, il se peut que, suite à ce biais, les prédictions d'interactions soient meilleures chez les organismes unicellulaires, et chez les organismes supérieurs pour les interactions mettant en cause des gènes de ménage (house-keeping genes), plutôt que pour des gènes impliqués dans la différenciation cellulaire.

4.3.b Mise à jour de InterDB

Les logiciels développés pour la construction de la base InterDB se révèlent très efficaces. Ainsi, la dernière reconstruction complète de InterDB (mise à jour majeure, voir ci-dessous) a pris 30 minutes, pour traiter 1.1 Go de fichiers source textuels. Par exemple, l'analyse syntaxique des 850 Mo de fichiers SwissProt et TrEMBL prend 12 minutes. Au total, 300 Mo de fichiers au format *.ace* sont générés et lus dans une base acedb vierge. La machine utilisée est un ordinateur PC sous Linux, comportant un processeur PentiumII 400MHz et 256 Mo de mémoire.

Nous pouvons donc les utiliser pour mettre la base à jour facilement. Deux types de mises à jour sont envisageables : les majeures où l'on va reconstruire une

base InterDB *ex nihilo*, et les mineures, où l'on va enrichir une base existante en y ajoutant par exemple de nouvelles interactions.

Les mises à jour majeures correspondent à l'utilisation d'une nouvelle version de SwissProt et TrEMBL. La version actuelle de InterDB est la troisième reconstruction majeure. Une telle opération présente deux avantages. D'une part, elle permet de bénéficier d'annotations de meilleure qualité, notamment pour les mots-clés et la localisation cellulaire, en exploitant les corrections apportées par les responsables de SwissProt et TrEMBL. D'autre part, elle augmente le nombre de protéines dans InterDB, ce qui permet de reconnaître plus d'interacteurs et donc d'écarter moins d'interactions provenant de nos bases source.

Les mises à jour mineures sont celles où une base InterDB existante est enrichie : le squelette protéique de la base reste inchangé, mais de nouvelles interactions ou de nouveaux descripteurs de protéines sont intégrés. Par exemple, les domaines et familles de la base Interpro ont pu être ajoutés à InterDB de cette manière. De même, les interactions découvertes en collaboration avec Anne Davy [Dav01] ont été intégrées ainsi. L'intérêt de ces mises à jour mineures est qu'elles sont plus légères à mettre en œuvre que les majeures, et qu'elles permettent de conserver un référentiel protéique constant tout en enrichissant la base. Ce dernier point est important dans la mesure où nos travaux sur la prédiction d'interactions entre protéines utilisent les données d'InterDB, mais reformatées sous une forme adaptée aux méthodes informatiques employées. Il est important de conserver une cohérence entre ces deux formes, particulièrement lors de la mise au point des algorithmes prédictifs, mais il peut tout de même être intéressant d'enrichir la base au cours de ces travaux. Dans ce cas une mise à jour mineure s'impose.

Il est intéressant de remarquer que le fichier SwissProt n'est pas toujours conforme à la spécification de la documentation : notre interpréteur a détecté un ou plusieurs problèmes syntaxiques sur chaque version de SwissProt+TrEMBL utilisée. Ainsi, à l'occasion de chacune des trois mises à jour majeures de InterDB, nous avons pu soumettre des corrections aux responsables de ces bases, corrigeant des erreurs purement syntaxiques mais néanmoins handicapantes pour toute analyse automatique.

4.4 Conclusion

Nous avons sélectionné deux bases de données de protéines, SwissProt et TrEMBL, qui sont complémentaires et semblent bénéficier d'annotations de bonne qualité. Nous avons identifié trois sources de données d'interactions protéine-protéine déterminées expérimentalement : WISTdb, YPD et DIP. Nous avons recherché des sources de caractérisations fonctionnelles ou structurales des protéines, susceptibles d'expliquer les interactions observées et donc d'en prédire de nouvelles. Dans cette optique, nous nous sommes concentrés sur les descripteurs disponibles dans les bases de motifs, domaines et familles protéiques, telles que ProSite et Pfam, et avons migré dès que possible vers la base fédératrice Interpro. Nous exploitons également certaines annotations provenant de SwissProt et TrEMBL : les mots-clés, et les localisations cellulaires.

En intégrant toutes ces sources de données, nous avons construit une base de données multi-organisme d'interactions entre protéines orientée prédiction, nommée InterDB. Cette base contient actuellement 2464 interactions concernant 2032 protéines, provenant principalement de la levure *S. cerevisiae*, de l'Homme et du nématode *C. elegans*.

Pour construire InterDB, nous avons développé un système logiciel composé d'un ensemble d'interpréteurs pour les bases d'origine. Ces interpréteurs produisent une représentation des données au format *.ace*, conformément au schéma choisi pour InterDB. Notre système permet de mettre facilement InterDB à jour lorsque de nouvelles données deviennent disponibles dans les bases source.

Ce travail était relativement technique mais nécessaire. Je l'ai accompli avec plaisir et j'en ai retiré une certaine maîtrise du langage Perl et de l'interface AcePerl, qui m'ont servi par la suite (voir partie 5) et me serviront certainement dans le futur.

Les perspectives de développement d'InterDB sont multiples. Au delà de la maintenance de la base, j'envisage d'incorporer des interactions provenant de sources récentes et prometteuses, en particulier la base BIND [Bad01] et les travaux de Ito *et al* [Ito01]. D'autre part, je souhaite intégrer de nouveaux descripteurs, découlant des travaux du Gene Ontology Consortium [Ash00] qui

propose une classification hiérarchique des gènes de la levure, de la souris, de l'Homme, et bientôt de *C. elegans*.

5. Prédiction d'interactions protéine-protéine

La base de données InterDB (voir chapitre 4) recense des interactions entre protéines. Chaque protéine y est représentée par un ensemble de descripteurs. Nous avons choisi ces descripteurs dans l'espoir qu'ils puissent expliquer les interactions observées, et donc permettre de caractériser et définir des classes d'interactions entre protéines. Il s'agit de rechercher des affinités entre descripteurs, sous forme de règles telles que : une protéine décrite par les descripteurs D1 et D2 est susceptible d'interagir avec toute protéine décrite par les descripteurs D'3, D'4 et D'5. Afin d'induire de telles règles, nous avons développé un système d'extraction de connaissances qui repose sur des algorithmes de Fouille de Données Relationnelles (Datamining), dits de recherche d'ensembles fréquents.

Comme dans tout processus d'extraction de connaissances, plusieurs tâches doivent être accomplies avant et après application des algorithmes de Datamining. En partant d'une base de données propre comme InterDB, il est tout d'abord nécessaire de modéliser les données sous une forme appropriée. Il devient alors possible d'appliquer l'algorithme choisi. Il est ensuite crucial de concevoir et de mettre en œuvre des filtres de post-traitement, afin de trier, d'évaluer et d'interpréter les règles induites. Enfin, une dernière tâche consiste à exploiter les règles retenues, en l'occurrence dans le cadre d'un logiciel de prédiction d'interactions entre protéines.

Dans ce chapitre, nous exposons successivement ces quatre composants de notre système d'extraction de connaissances. Nous présentons finalement les expérimentations menées et les résultats obtenus.

5.1 Modélisation des données

Rappelons que nous disposons dans InterDB de 2464 interactions concernant 2032 protéines (voir 4.3.a). Ces protéines sont caractérisées par 763 domaines Interpro, 374 mots-clefs SwissProt et 33 descripteurs de localisation cellulaire, pour un total de 1170 descripteurs.

Pour appliquer les algorithmes de recherche d'ensembles fréquents, il est tout d'abord nécessaire de modéliser les données étudiées sous une forme appropriée, à savoir sous forme d'une matrice booléenne. Dans cette matrice, les colonnes doivent représenter des traits observables et les lignes doivent représenter des observations. L'algorithme de recherche d'ensembles fréquents permet alors d'identifier les ensembles de traits qui sont fréquemment observés ensemble, autrement dit les ensembles de traits qui sont tous vrais dans au minimum un nombre fixé d'observations (voir 5.2.a). La modélisation choisie est la suivante.

Nous construisons une matrice booléenne dans laquelle chaque ligne représente une interaction entre deux protéines et chaque colonne un descripteur. Toute interaction impliquant deux protéines, la matrice comprend 2340 (1170 x 2) colonnes et 4928 (2464 x 2) lignes. Les 1170 premières colonnes représentent les descripteurs qui caractérisent la première protéine, soit protéine 1, alors que les 1170 colonnes suivantes correspondent aux descripteurs de la seconde protéine, soit protéine 2. Pour exploiter la symétrie inhérente de la relation binaire *interaction*, chaque interaction est représentée par deux lignes, en choisissant l'un ou l'autre des interacteurs comme protéine 1. Ainsi, pour chaque ensemble fréquent dans cette matrice, il existe un ensemble fréquent dual de fréquence identique, qui représente en fait la même association entre descripteurs.

Il se présente une difficulté dans le cas des ensembles fréquents symétriques, c'est-à-dire des ensembles pour lesquels les deux protéines sont caractérisées par les mêmes descripteurs. En effet, ces ensembles apparaissent à tort comme deux fois plus fréquents que les ensembles dissymétriques. Remarquons que ce problème n'est pas une conséquence du choix de représenter les interactions sur deux lignes : il est simplement dû à la symétrie inhérente de la

relation "interaction". Ce problème ainsi que la solution proposée sont repris et décrits en détail dans la partie 5.3.a.3.

Nous avons écrit un script, assocrules.tablemaker.pl, pour effectuer le recensement des descripteurs utilisés et pour construire la matrice booléenne précitée. Comme la majorité des programmes développés dans le cadre de notre système d'extraction de connaissances pour la prédiction d'interactions protéine-protéine, il est écrit en Perl et exploite AcePerl [Ste99], une interface programmeur aux bases acedb en Perl.

Ce programme permet de définir des critères, sous forme de requêtes acedb, pour sélectionner les interactions qui seront utilisées pour l'apprentissage. Cette approche permet de réserver un sous-ensemble des interactions d'InterDB pour la validation des règles prédictives induites (voir 5.5.b).

5.2 Recherche d'ensembles fréquents

Nos données d'interaction étant ainsi modélisées sous la forme d'une matrice booléenne, elles deviennent exploitables par les algorithmes de recherche d'ensembles fréquents tels que *apriori* [Agr96] et *close* [Pas99]. Rappelons qu'un ensemble fréquent est un ensemble de colonnes de la matrice booléenne qui prennent la valeur *vrai* dans les même lignes, et ceci avec une fréquence minimale F_{min}, où F_{min} est le seuil de fréquence choisi (voir 2.2).

En collaboration avec l'équipe de Jean-François Boulicaut de l'INSA-Lyon, nous avons pu utiliser une implémentation de l'algorithme *min-ex* [Bou00], une variante de l'algorithme *close* développée au Laboratoire d'Ingénierie des Systèmes d'Information de l'INSA-Lyon.

Dans cette partie, nous présentons d'abord l'algorithme *min-ex* et décrivons en particulier la représentation condensée des ensembles fréquents qu'il exploite. Nous exposons ensuite sa mise en œuvre dans le cadre de notre système d'extraction de connaissances pour la prédiction d'interactions protéine-protéine.

5.2.a L'algorithme *min-ex*

L'algorithme *min-ex* [Bou00] a été développé et implémenté au sein de l'équipe de Jean-François Boulicaut, au Laboratoire d'Ingénierie des Systèmes d'Information de l'INSA-Lyon. Nous présentons d'abord le principe de l'algorithme dans ses grandes lignes, puis décrivons plus en détail la représentation condensée des ensembles fréquents qu'il exploite.

5.2.a.1 Principe de l'algorithme

L'algorithme *min-ex* permet de rechercher les ensembles fréquents dans une matrice booléenne. Il s'agit d'une variante de l'algorithme *close*.

Rappelons que *close* (voir 2.2.d.3) repose sur la notion de fermeture des ensembles fréquents. La fermeture d'un ensemble fréquent E peut être définie

comme le plus grand sur-ensemble de E dont la fréquence est égale à la fréquence de E. Une propriété intéressante de la fermeture peut s'énoncer comme suit. En notant pour tout ensemble X fermé(X) sa fermeture et fréquence(X) sa fréquence, on a pour tout ensemble E :

$\forall F, [E \subseteq F \subseteq \text{fermé}(E)] \Rightarrow \text{fréquence}(F) = \text{fréquence}(E) = \text{fréquence}(\text{fermé}(E))$.

L'algorithme *close* exploite cette propriété pour éviter de calculer les fréquences des ensembles qui sont compris entre un ensemble dont on a calculé la fréquence et le fermé de cet ensemble dans le treillis d'inclusion.

L'algorithme *min-ex* introduit la notion d'itemsets "presque-fermés". L'idée consiste à introduire un paramètre entier positif ou nul δ, qui représente la différence tolérée sur les supports.

Le δ-presque-fermé d'un ensemble E est le plus grand sur-ensemble de E dont le support est égal au support de E, à une erreur δ près. Ainsi, l'algorithme *min-ex* est équivalent à l'algorithme *close* dans le cas où l'on choisit $\delta = 0$. Avec $\delta > 0$, le δ-presque fermé d'un ensemble E est un sur-ensemble du fermé de E.

Le principe de l'algorithme est similaire à celui de *close*. Il consiste à éviter de calculer les fréquences des ensembles compris entre un ensemble utilisé comme générateur (voir 2.2.d.3) et son δ-presque-fermé. Avec $\delta > 0$, le nombre d'ensembles que l'on évite ainsi de considérer est plus important qu'avec *close*. Il s'ensuit que *min-ex* peut permettre de rechercher des ensembles fréquents avec un seuil de fréquence plus faible que *close*, au prix d'une légère imprécision sur les fréquences des ensembles.

Remarquons que dans nos expérimentations, nous avons finalement fixé le seuil d'erreur δ à 0. Notre utilisation de *min-ex* est donc fonctionnellement équivalente à une utilisation de *close*. Nous avons fait ce choix car avec $\delta=0$, nous pouvons déjà choisir un seuil de fréquence aussi petit que nécessaire. Il n'est donc pas utile dans ce cas de tolérer d'erreur dans le calcul des fréquences.

5.2.a.2 Représentation condensée des ensembles fréquents

Plutôt que de fournir la liste complète des ensembles fréquents, qui peut se révéler extrêmement longue étant donné que le seuil de fréquence peut être très faible, l'implémentation de *min-ex* utilisée met en œuvre une représentation

condensée des ensembles fréquents ([Man96], [Bou00]).

Cette représentation condensée permet de considérer un nombre très important d'ensembles fréquents, sans les désigner en extension. Elle repose sur le concept de fermeture, et exploite la propriété déjà énoncée en 5.2.a.1, qui stipule que la fréquence de tout ensemble contenant un ensemble E et contenu dans la fermeture de E est égale à la fréquence de E.

Un ensemble condensé est représenté sous la forme :

T, C : F

où T et C sont des ensembles de traits et F est un nombre. Nous dirons que T est la *tête* de l'ensemble condensé, C est son *corps*, et F sa fréquence. Cette notation signifie que le fermé de T est fermé(T) = T ∪ C. De plus, F est la fréquence de T. F est donc également la fréquence de tout ensemble G tel que T ⊆ G ⊆ (T ∪ C).

Avant d'illustrer cette définition par un exemple, il est important de préciser quelques notations qui seront utilisées dorénavant dans cet exposé. Rappelons que les traits (ou items) sont, dans notre application, des descripteurs qui peuvent caractériser soit la protéine 1, soit la protéine 2 (voir 5.1). Afin de faciliter la compréhension de certains exemples, il importe de distinguer ces deux ensembles de traits. Dans la suite, nous notons D1, D2, …, Dn les descripteurs concernant la protéine 1 et D'1, D'2, …, D'n ceux concernant la protéine 2.

Exemple :

Considérons l'ensemble condensé suivant : D1, D2 D'3 : F. Dans cette notation, D1 et D2 concernent donc la protéine 1 et D'3 concerne la protéine 2. F est la fréquence de tous les ensembles compris entre {D1} et {D1,D2,D'3} dans le treillis d'inclusion. En l'occurrence, F est donc la fréquence des ensembles suivants : {D1}, {D1,D2}, {D1,D'3}, {D1,D2,D'3}. D'un point de vue logique, l'ensemble condensé représente que D1⇒D2∧D'3 : tout ensemble contenant D1 contient aussi D2 et D'3.

5.2.b Utilisation de *min-ex*

Une fois les données d'interaction modélisées sous forme d'une matrice booléenne, nous pouvons appliquer le programme *min-ex*. Nous obtenons ainsi une liste d'ensembles condensés fréquents extraits de la matrice. Il s'agit des

ensembles condensés dont la fréquence est supérieure à un seuil de fréquence fixé par l'utilisateur. Rappelons que l'objectif final est de produire des règles prédictives, sous la forme d'associations entre ensembles de descripteurs. Le seuil de fréquence joue un rôle déterminant quant à la qualité des règles induites.

Nous discutons d'abord dans cette section du choix du seuil de fréquence. Nous expliquons ensuite l'approche retenue pour exploiter les ensembles condensés extraits.

5.2.b.1 Choix du seuil de fréquence

L'implémentation de *min-ex* à laquelle nous avons eu accès s'est révélée très efficace. En effet, elle permet d'extraire les ensembles condensés fréquents de nos données en quelques minutes, même en choisissant un seuil de fréquence extrêmement bas. De plus, elle fournit les ensembles fréquents sous une représentation condensée (voir 5.2.a.2).

Ces qualités nous permettent de considérer un nombre très important d'ensembles. Nous avons choisi de fixer le seuil de fréquence à 0.0004, de manière à considérer tout ensemble disposant d'un support de 2 comme fréquent. Une valeur aussi faible signifie que nous produisons beaucoup de bruit, c'est-à-dire que de nombreux ensembles apparemment fréquents ne représentent pas réellement des affinités entre descripteurs. Par contre, ceci nous permet de produire des règles bien plus intéressantes qu'avec un seuil plus élevé, car dans ce dernier cas les règles obtenues seraient certainement plus sûres mais souvent triviales. Il s'ensuit que notre système dépend fortement des post-traitements, qui doivent distinguer efficacement les règles prometteuses des artéfacts.

5.2.b.2 Utilisation des ensembles fréquents maximaux

Rappelons que l'ensemble condensé T, C : F représente tous les ensembles contenant T et contenus dans (T $\cup$ C). Si nous déterminons à l'aide de nos filtres de post-traitement que l'ensemble condensé est significatif, la question se pose de savoir comment il doit être exploité.

Une possibilité consiste à produire une règle, sous forme d'association entre ensembles de descripteurs, correspondant à chaque ensemble fréquent

représenté par cet ensemble condensé. Cependant, dans ce cas, la règle qui correspond à T est une généralisation de toutes les autres règles induites, qui peuvent donc être éliminées. Cette stratégie revient donc à ne retenir que la règle la plus générale qu'on peut extraire de chaque ensemble condensé.

Nous ne souhaitons pas adopter cette solution, car dans notre contexte il nous paraît plus intéressant d'obtenir les règles les plus spécifiques possible. Nous avons choisi de conserver uniquement la règle extraite de l'ensemble fréquent *maximal*, au sens de l'inclusion, représenté par chaque ensemble condensé. En considérant un ensemble condensé T, C : F, il s'agit en fait du fermé de T, à savoir $(T \cup C)$.

5.3 Post-traitements

Rappelons que l'algorithme *min-ex* permet d'utiliser un seuil de fréquence extrêmement faible (voir 5.2.b.1). Il s'ensuit que notre système dépend fortement des post-traitements, qui doivent distinguer efficacement les règles prometteuses des artéfacts. Nous avons donc conçu et mis en œuvre quatre filtres de post-traitement, afin d'écarter les ensembles fréquents non-significatifs. En pratique, l'utilisation de la représentation condensée des ensembles fréquents (voir 5.2.a.2) introduit certaines difficultés techniques supplémentaires pour le développement des post-traitements. Ces difficultés ne sont pas spécifiques de notre domaine d'application. A ce titre, nous les décrivons en détail, ainsi que les solutions proposées.

5.3.a Objectif et méthode

Dans la suite, nous présentons les quatre filtres de post-traitement que nous avons conçus pour écarter les ensembles fréquents non significatifs ou sans intérêt pour la prédiction d'interactions entre protéines. Nous insistons particulièrement sur les aspects liés à l'utilisation de la représentation condensée des ensembles fréquents.

Ces quatre filtres de post-traitement sont présentés successivement et concernent :

- Les ensembles fréquents qui contiennent un descripteur non-convenable ou sans rapport avec les interactions entre protéines ;
- Les ensembles fréquents qui ne concernent qu'une seule protéine ;
- Les ensembles fréquents non-significatifs selon une mesure que nous définissons ;
- Les ensembles fréquents qui sont sous-ensembles d'autres ensembles fréquents.

5.3.a.1 Elimination d'ensembles fréquents contenant un descripteur non-convenable

Certains descripteurs ne présentent pas d'intérêt pour l'étude des interactions entre protéines. Typiquement, c'est le cas du mot-clé SwissProt *Hypothetical_Protein*, qui signifie que l'existence de la protéine considérée n'est pas confirmée expérimentalement. Ce mot-clé n'a évidemment aucun rapport avec les interactions entre protéines. Il en ressort que les ensembles qui le contiennent doivent être écartés.

Conceptuellement, ces "mauvais" descripteurs auraient pu être éliminés au cours de la phase de modélisation des données, c'est-à-dire avant application de l'algorithme *min-ex*. Cependant la vérification initiale de tous les descripteurs est une tâche ardue. En effet, les significations des descripteurs sont souvent obscures et nécessitent une étude approfondie. Il est bien plus facile de vérifier a posteriori que les descripteurs présents dans un ensemble fréquent prometteur peuvent raisonnablement jouer un rôle dans les interactions entre protéines. De plus, cette approche permet de travailler plus efficacement avec nos collaborateurs biologistes : s'ils nous informent, au cours d'un examen des règles, qu'un descripteur ne leur paraît pas approprié, il suffit de ré-appliquer les filtres de post-traitement en tenant compte de leur remarque. Comme ces filtres sont mis en œuvre de manière extrêmement efficace, cette stratégie est bien plus avantageuse que de ré-appliquer *min-ex* sur un jeu de données corrigées, d'autant plus que *min-ex* n'est pas disponible hors site.

Dans le cas des ensembles fréquents condensés (voir 5.2.a.2), on ne peut se contenter d'éliminer tout ensemble condensé contenant un "mauvais" descripteur. En effet, deux cas de figure doivent être considérés pour traiter un ensemble condensé contenant un "mauvais" descripteur, selon que celui-ci apparaît dans la tête ou dans le corps de l'ensemble condensé. S'il apparaît dans la tête, il est en fait présent dans tout ensemble fréquent représenté par l'ensemble condensé. L'ensemble condensé au complet doit donc être éliminé. Par contre, s'il apparaît dans le corps, certains ensembles fréquents représentés par l'ensemble condensé ne contiennent pas ce "mauvais" descripteur, et ne doivent pas être écartés. Le traitement correct consiste à conserver l'ensemble condensé en retirant seulement le "mauvais" descripteur de son corps.

Exemple :

Soient D1, D2, D3 et D4 des descripteurs.

Supposons que D1 soit un "mauvais" descripteur. L'ensemble condensé D1 D2, D3 D4 : F représente l'ensemble des ensembles contenant D1 et D2, et contenus dans {D1 D2 D3 D4}, à savoir : {D1 D2}, {D1 D2 D3}, {D1 D2 D4} et {D1 D2 D3 D4}. Ces ensembles contiennent tous D1 et doivent donc être écartés. Dans ce cas, nous éliminons l'ensemble condensé complet.

Supposons maintenant que D3 soit un "mauvais" descripteur, et considérons toujours l'ensemble condensé D1 D2, D3 D4 : F. Cette fois les ensembles {D1 D2 D3} et {D1 D2 D3 D4} doivent être certes écartés, mais {D1 D2} et {D1 D2 D4} doivent être conservés. Nous remplaçons dans ce cas l'ensemble condensé initial par D1 D2, D4 : F. Il s'agit bien de l'ensemble condensé qui représente exclusivement les ensembles à conserver.

5.3.a.2 Elimination d'ensembles fréquents ne concernant qu'une seule protéine

Certains ensembles fréquents ne concernent qu'une seule protéine. Ces ensembles correspondent en fait à des descripteurs liés. Il s'agit souvent de synonymes. Par exemple, le domaine SH2 est représenté à la fois par un domaine Interpro et par le mot-clé *SH2_Domain* dans SwissProt. Ces ensembles peuvent avoir une signification bien réelle, mais ils ne peuvent servir à prédire des interactions entre protéines. Ils sont donc écartés.

En pratique, un ensemble fréquent condensé est dans cette classe si tous les descripteurs de sa tête et de son corps correspondent à une seule protéine. Remarquons qu'il est important d'effectuer ce filtre après le précédent (voir 5.3.a.1), car certains ensembles fréquents condensés peuvent migrer vers cette classe d'ensembles indésirables après élimination des "mauvais" descripteurs.

Exemple :

Considérons l'ensemble condensé suivant : D1, D2 D'3 : F. Rappelons que dans cette notation, D1 et D2 concernent la protéine 1, et D'3 concerne la protéine 2. Supposons que D'3 soit un "mauvais" descripteur. Après application du filtre précédent, l'ensemble condensé devient D1, D2 : F. Cet ensemble condensé ne concerne que la protéine 1, et est donc écarté par notre second filtre.

5.3.a.3 Notion de score pour les ensembles fréquents, seuil de score éliminatoire

Nous proposons un score de significativité pour chaque ensemble fréquent, constitué comme suit. Considérons un ensemble fréquent E, observé avec la fréquence F. On peut poser E = {D1 … Dn D'1 … D'p}, où D1 à Dn sont des descripteurs de la protéine 1, et D'1 à D'p décrivent la protéine 2. Les ensembles {D1 … Dn} et {D'1 … D'p} sont évidemment fréquents, observés avec les fréquences respectives F1 et F2. En l'absence de toute affinité entre les ensembles de descripteurs {D1 … Dn} et {D'1 … D'p}, on s'attend à observer l'ensemble E avec la fréquence (F1xF2). Le score proposé pour l'ensemble E est le rapport entre sa fréquence observée et sa fréquence attendue, c'est-à-dire : F/(F1xF2).

Remarquons que ce score n'est pas valide pour les ensembles fréquents symétriques, c'est-à-dire tels que les descripteurs des protéines 1 et 2 sont identiques. Pour ces ensembles, une correction doit être apportée en divisant le score par deux, pour obtenir F/(2xF1xF2).

Exemple :

Pour illustrer ce dernier point et se convaincre de son bien-fondé, considérons les 3 interactions suivantes :

(D1,D'2),

(D2,D'1),

(D3 D4,D'3).

Ces notations signifient que : dans la première interaction, la protéine 1 est décrite par le descripteur D1 et la protéine 2 par le descripteur D2 ; dans la seconde interaction, la protéine 1 est décrite par D2 et la protéine 2 par D1 ; et dans la troisième interaction la protéine 1 est décrite par les descripteurs D3 et D4 et la protéine 2 est décrite par D3. Au vu de ces 3 interactions, l'affinité entre les descripteurs D1 et D2 est 2 fois plus importante qu'entre D3 et lui-même, puisque cette affinité pourrait expliquer les deux premières interactions alors que l'affinité D3-D3 n'explique que la dernière interaction observée. Cependant, ceci n'apparaît pas au vu des valeurs des fréquences attendues des ensembles {D1 D'2}, {D2 D'1} et {D3 D'3}, qui apparaissent toutes identiques. La division du score par deux pour les ensembles symétriques, dans ce cas {D3 D'3}, permet de corriger cette erreur.

Le troisième filtre que nous proposons procède en calculant les scores des

ensembles fréquents maximaux, et en éliminant tous les ensembles dont le score est inférieur à un seuil fourni par l'utilisateur. Rappelons que pour chaque ensemble condensé, nous avons choisi de ne conserver que l'ensemble fréquent maximal (voir 5.2.b.2).

Pour déterminer le score d'un ensemble fréquent E, il est nécessaire de connaître la fréquence observée des sous-ensembles de E qui concernent exclusivement la première ou la seconde protéine, respectivement notés E1 et E2. Comme ces ensembles sont eux-mêmes fréquents, leurs fréquences ont déjà été calculées par l'algorithme de Datamining utilisé. Cependant, dans le cas de l'algorithme *min-ex*, nous avons simplement la garantie qu'il existe un ensemble fréquent condensé qui représente chacun des ensembles E1 et E2. Pour calculer le score de E, il est donc nécessaire d'identifier les ensembles condensés représentant E1 et E2. A cette fin, nous avons développé un algorithme (voir 5.3.b.1) qui repose sur l'observation suivante.

Un ensemble condensé T , C : F (T est la tête, C le corps et F la fréquence) représente l'ensemble E1 si et seulement si $T \subseteq E1$ et $R \subseteq C$, où R est le complément de T dans E1. Si un tel ensemble est trouvé, on peut déterminer la fréquence de E1, qui vaut F.

Remarquons que la fréquence de E1 est unique : il est possible que plusieurs ensembles condensés représentent E1, mais dans ce cas les fréquences de ces ensembles condensés sont par définition toutes égales.

Exemple :

Soit E1 = {D1 D2} un ensemble fréquent dont on cherche à déterminer la fréquence. Supposons que *min-ex* ait identifié les ensembles condensés fréquents suivants. Précisons que cet exemple est construit afin d'illustrer le calcul de la fréquence de E1, mais qu'il ne correspond pas à une situation réellement envisageable. En effet, ces ensembles condensés ne sont pas cohérents entre eux.

D1 D3, D2 : F1

D1 D2 D3, : F2

D1, D3 D4 : F3

D1 D2, D3 : F4

D1, D2 D3 : F5

Le premier ensemble condensé ne représente pas E1, car D3 est dans sa tête. Le second ensemble condensé ne représente pas E1 non plus, pour la même

raison. Remarquons que la présence de D1 dans la tête et de D2 dans le corps (premier ensemble) ou dans la tête (second ensemble) ne change rien : tout ensemble condensé dont la tête n'est pas incluse dans E1 ne peut en aucun cas représenter E1.

Considérons maintenant les trois derniers ensembles condensés. Ils satisfont tous la première contrainte : leurs têtes, respectivement {D1}, {D1 D2} et {D1}, sont toutes incluses dans E1.

Cependant le corps du troisième ensemble condensé ne contient pas D2. Cet ensemble ne représente donc pas E1. Par contre les quatrième et cinquième ensembles condensés satisfont la seconde contrainte. En effet, dans le quatrième ensemble condensé le complément de la tête dans E1 est ∅, et dans le cinquième le complément de la tête dans E1 est {D2}, qui est inclus dans le corps {D2 D3} de l'ensemble condensé.

Finalement, la fréquence de E1 est donc F4 et F5. On peut également déduire que F4=F5.

5.3.a.4 Elimination de sous-ensembles d'ensembles fréquents

Nous avons conçu le quatrième filtre dans le souci de favoriser les règles spécifiques, par rapport à des règles plus générales dont le score serait du même ordre. Rappelons que c'est aussi l'objectif de notre stratégie qui consiste à ne retenir que l'ensemble fréquent maximal correspondant à chaque ensemble condensé (voir 5.2.b.2). Pour illustrer cette notion de spécificité, considérons l'exemple suivant.

Exemple :

Considérons les deux ensembles fréquents E1 = {D1 D'4} et E2 = {D1 D2 D3 D'4 D'5}, de scores respectifs S1 et S2. Rappelons que E1 est interprété comme la règle R1 suivante : toute protéine décrite par D1 est susceptible d'interagir avec toute protéine décrite par D4, et que E2 est interprété comme la règle R2 : toute protéine décrite par D1, D2 et D3 est susceptible d'interagir avec toute protéine décrite par D4 et D5.

Il est clair que R2 est plus spécifique que R1. Si les deux règles étaient retenues, R2 ne pourrait prédire que des interactions potentielles prédites également par R1. Par contre, de nombreuses interactions prédites par R1 ne le

seraient pas par R2. Il apparaît donc utile de ne retenir que l'une ou l'autre de ces deux règles.

D'une manière générale, soient E1 et E2 deux ensembles fréquents tels que $E1 \subset E2$, et soient R1 et R2 les règles correspondantes. R2 est donc une spécialisation de R1. Il serait a priori souhaitable d'éliminer la règle R1 afin de ne retenir que la règle plus spécifique R2. Néanmoins, certaines règles générales peuvent s'avérer très bonnes, notamment si les descripteurs qui y apparaissent (dans l'exemple précédent, D1 et D'4) présentent une forte affinité. Il est donc nécessaire de définir un critère qui permette de choisir entre R1 et R2.

Nous proposons un critère simple qui tient compte des scores respectifs des règles, et qui est paramétrable par l'utilisateur bioinformaticien. La règle plus générale R1 est éliminée si son score S1 est plus faible, à un facteur multiplicatif près fourni en paramètre, que le score S2 de la règle plus spécifique R2.

Exemple :

En reprenant l'exemple précédent, considérons les deux ensembles fréquents E1 = {D1 D'4} et E2 = {D1 D2 D3 D'4 D'5}, de scores respectifs S1 et S2. Soit K la valeur du facteur multiplicatif fourni par l'utilisateur. Si $S1 < (S2 \times K)$, alors E1 est éliminé et seule la règle plus spécifique correspondant à E2 est conservée. Sinon, les deux règles sont conservées.

5.3.b Mise en œuvre

Nous exposons dans cette partie la mise en œuvre des quatre filtres de post-traitement conçus (voir 5.3.a). Pour des raisons d'efficacité, les trois premiers filtres sont mis en œuvre dans un premier programme, nommé filtre.minex.pl, et le quatrième filtre est implémenté dans un second programme nommé filtre.2.minex.iter.pl. Ces deux programmes, assortis de commentaires, sont présentés en annexes 5.1 et 5.2, respectivement.

Nous présentons pour chacun de ces programmes les algorithmes développés, et étudions les complexités associées. Nous montrons en particulier que ces algorithmes sont linéaires en temps, en fonction du nombre total d'ensembles fréquents à traiter.

5.3.b.1 Mise en œuvre des trois premiers filtres de post-traitement

L'algorithme mis en œuvre traite les ensembles fréquents condensés produits par *min-ex* en deux passes successives. Dans les grandes lignes, cet algorithme est le suivant. Dans une première passe, les deux premiers filtres (voir 5.3.a.1 et 5.3.a.2) sont appliqués pour produire un fichier temporaire T, contenant des ensembles fréquents maximaux. De plus, une table de hash H est peuplée. H est appelée à contenir tous les ensembles condensés qui représentent au moins un ensemble fréquent concernant une seule protéine. Dans une seconde passe, le fichier temporaire T est lu et la table de hash H permet de calculer le score de chaque ensemble fréquent maximal de T, permettant ainsi d'appliquer le troisième filtre (voir 5.3.a.3). Dans la suite, nous présentons plus en détail l'algorithme en précisant les difficultés rencontrées et les solutions proposées, et nous calculons sa complexité.

Dans la première passe, le premier filtre est tout d'abord appliqué. Ensuite, les ensembles condensés qui représentent au moins un ensemble fréquent concernant une seule protéine sont stockés dans la table de hash H, indexée par la tête des ensembles condensés. Il s'agit en fait des ensembles condensés dont la tête ne concerne qu'une seule protéine.

Précisons que les ensembles fréquents ainsi que les têtes et corps des ensembles condensés sont systématiquement représentés par des listes triées. D'autre part, par construction (voir 5.2.a.2) il ne peut exister qu'un seul ensemble condensé ayant pour tête un ensemble donné. Il est ainsi possible d'utiliser les têtes des ensembles condensés comme index dans la table de hash H.

Les ensembles condensés représentant au moins un ensemble qui concerne les deux protéines, c'est-à-dire les ensembles condensés acceptés par le second filtre, sont quant à eux stockés dans le fichier temporaire T qui sert de fichier d'entrée pour la passe 2. Un ensemble condensé peut donc apparaître à la fois dans la table de hash H et dans le fichier temporaire T.

Exemple :

Considérons l'ensemble condensé EC suivant :

D1, D2 D'3 : F.

Cet ensemble condensé EC représente les ensembles fréquents {D1} et {D1 D2}

qui ne concernent que la première protéine. EC est donc stocké dans la table de hash H. Cependant, EC représente aussi les ensembles fréquents {D1 D'3} et {D1 D2 D'3} qui concernent les deux protéines. L'ensemble condensé EC est donc également stocké dans le fichier temporaire T.

Dans la seconde passe, le fichier temporaire T est lu pour l'application du troisième post-traitement. Ce fichier contient en fait les ensembles fréquents maximaux résultants de l'application des deux premiers filtres aux données produites par *min-ex*. Pour chaque ensemble fréquent maximal E, les sous-ensembles E1 et E2, qui concernent respectivement la première et la seconde protéine, sont construits. Leurs fréquences respectives F1 et F2 sont obtenues à l'aide de l'algorithme décrit ci-dessous. Le score de E est alors calculé. Si ce score est inférieur au seuil spécifié, E est écarté.

Etant donné un ensemble E1 que l'on sait fréquent, l'idée pour obtenir sa fréquence est la suivante. Nous construisons successivement tous les sous-ensembles S de E1. Pour chaque S, nous recherchons dans la table de hash H un ensemble condensé dont la tête est S. Si un tel ensemble condensé EC existe, nous déterminons si son corps contient le complément de S dans E1. Si c'est le cas, la fréquence de E1 est la fréquence de l'ensemble condensé EC. Sinon, nous considérons le sous-ensemble suivant de E1.

Dans le pire cas, cet algorithme est de complexité $2^{|E1|}$ x 1 x |E1|. En effet :
- le facteur $2^{|E1|}$ correspond au nombre de sous-ensembles de E1 ;
- le facteur 1 correspond à la recherche d'un sous-ensemble de E1, en tant que clé, dans la table de hash H ;
- le facteur |E1| correspond à la recherche du complément de S dans E1, étant donné que les ensembles sont représentés par des listes triées.

Or, la taille de E1 est bornée par le nombre maximum de descripteurs qui décrivent une protéine (inférieur à 20 dans la version actuelle de InterDB). L'algorithme de calcul du score d'un ensemble fréquent est donc en O(1) : il est en temps constant, bien que la constante soit assez importante.

Il s'ensuit finalement que le programme qui applique les trois premiers filtres est linéaire en fonction du nombre d'ensembles fréquents condensés présents dans le fichier produit par *min-ex*. Cette complexité théorique se traduit en pratique par des temps d'exécution de 40 secondes en moyenne pour traiter un

fichier comportant 125744 ensembles fréquents condensés (voir 5.5.a.3).

5.3.b.2 Mise en œuvre du quatrième filtre de post-traitement

En pratique, le quatrième filtre est appliqué au fichier produit par les filtres 1 à 3 (voir 5.3.b.1) qui contient des ensembles fréquents, et non pas des ensembles condensés. Sa mise en œuvre ne présente donc pas de difficultés particulières dues à l'utilisation de la représentation condensée. L'algorithme conçu est le suivant.

La liste des ensembles fréquents est tout d'abord triée par nombre croissant de descripteurs. Les ensembles fréquents sont ensuite considérés l'un après l'autre. Pour chaque ensemble fréquent, on construit successivement tous ses sous-ensembles. Chaque sous-ensemble est ensuite recherché dans une table de hash H' qui contient les ensembles fréquents valides. S'il est trouvé et qu'il ne remplit pas le critère sur les scores, il est éliminé de la table H'. L'ensemble fréquent initial est finalement lui-même stocké dans la table de hash H', qui est indexée par la liste triée représentant l'ensemble.

Le coût de la recherche du score d'un sous-ensemble pourrait poser problème. On peut cependant noter les deux points suivants. D'une part, grâce au tri initial des ensembles par nombre croissant de descripteurs, on est assuré que lorsqu'un ensemble E est considéré, tout ensemble susceptible de représenter une règle plus générale, c'est-à-dire tout sous-ensemble de E, a déjà été examiné. D'autre part, le choix de représenter chaque ensemble fréquent par une liste triée nous permet de les utiliser comme clés dans la table de hash. Ainsi, la recherche du score d'un sous-ensemble est effectuée en temps constant.

Enfin, on peut remarquer que le nombre de sous-ensembles à générer, pour chaque ensemble fréquent considéré, est borné par 2^N, où N est la taille du plus grand ensemble fréquent (égal à 36 en choisissant le seuil de fréquence extrêmement faible 0.0004, voir 5.2.b.1).

Il en ressort que l'algorithme appliquant le quatrième filtre est linéaire en fonction du nombre d'ensembles fréquents à considérer. Cette complexité théorique se traduit en pratique par des temps d'exécution inférieurs à 4 secondes en moyenne dans nos expérimentations, qui portent sur un fichier comportant initialement 125744 ensembles condensés (voir 5.5.a.3).

5.4 Détermination de prédictions d'interactions entre protéines

Pour chaque jeu de valeurs des paramètres de post-traitement, une classe d'ensembles fréquents que l'on espère significatifs est obtenue. Une règle prédictive peut être déduite de chacun de ces ensembles. Ces règles sont intégrées dans InterDB. Etant donné une protéine d'intérêt, il devient alors possible de proposer un ensemble de protéines qui sont susceptibles d'interagir avec cette protéine d'intérêt, en appliquant une ou plusieurs classes de règles prédictives. Le problème posé concerne la production et l'intégration dans InterDB des règles prédictives d'un part, et l'application de ces règles d'autre part.

5.4.a Production et intégration dans InterDB des règles prédictives

A tout ensemble fréquent significatif, on peut associer une règle prédictive. En effet, tout ensemble fréquent peut être représenté sous la forme {D1…Dn D'p…D'q}, où les Di sont des descripteurs de la protéine 1 et les D'j des descripteurs de la protéine 2. Cet ensemble signifie que l'on observe souvent et de manière significative une protéine décrite par D1, …, Dn en interaction avec une protéine décrite par D'p, …, D'q. Notre démarche consiste à induire de cette observation la règle suivante : toute protéine décrite par les descripteurs D1, …, Dn est susceptible d'interagir avec toute protéine décrite par les descripteurs D'p, …, D'q.

Ces règles sont hypothétiques, et certaines inspirent plus confiance que d'autres. D'une manière générale, les valeurs choisies pour les paramètres des post-traitements, c'est-à-dire la classe à laquelle appartient la règle, déterminent cette confiance. Cependant, cette confiance peut varier selon les règles, au sein même d'une classe de règles prédictives. Elle dépend en particulier du score de l'ensemble fréquent à partir duquel la règle a été induite (voir 5.3.a.3). Ce score est donc affecté comme score de confiance à chaque règle.

Ces règles sont ensuite intégrées dans la base InterDB. Un programme en

Perl permet de construire des fichiers au format *.ace* à partir des fichiers qui représentent les règles prédictives résultant de l'application des filtres de post-traitement. Dans la base de données InterDB, la classe ?Prediction permet de représenter les règles prédictives. Une autre classe nommée ?PredictedInteraction permet de représenter les interactions prédites (voir 5.4.b). Un extrait du fichier de définition du modèle de InterDB, représentant ces deux classes, est reproduit figure 5.1. Le modèle complet commenté est disponible en annexe 4.1.

```
?Prediction         Protein1            KeywordProt1 ?Keyword XREF Predictions
                                        Interpro1 ?Interpro XREF Predictions
                                        Localisation1 ?Localisation XREF Predictions
                    Protein2            KeywordProt2 ?Keyword XREF Predictions
                                        Interpro2 ?Interpro XREF Predictions
                                        Localisation2 ?Localisation XREF Predictions
                    AssociationRules        ObservedFreq UNIQUE Float
                                            ExpectedFreq UNIQUE Float
                                            FobsFexpMinRatio UNIQUE Float
                                            SubsetCutoffRatio UNIQUE Float
                                            BadKeywordCutoff UNIQUE Int

?PredictedInteraction   Protein1 UNIQUE ?Protein XREF PredictedInteractions
                        Protein2 UNIQUE ?Protein XREF PredictedInteractions
                        Valid ?Interaction XREF Predicted
                        Prediction ?Prediction
```

Figure 5.1 : Extrait du fichier models.wrm de définition du schéma de InterDB.

La classe ?Prediction possède les attributs KeywordProt1, Interpro1 et Localisation1 pour représenter les descripteurs de la protéine 1 (respectivement les mots-clés SwissProt, les domaines Interpro et les localisations cellulaires). De même, cette classe possède les attributs KeywordProt2, Interpro2 et Localisation2 pour représenter les descripteurs de la protéine 2. Enfin, elle possède les attributs ObservedFreq, ExpectedFreq, FobsFexpMinRatio SubsetCutoffRatio et BadKeywordCutoff pour stocker les fréquences observées et attendues de l'ensemble fréquent correspondant, ainsi que les valeurs des paramètres de post-

traitement utilisées.

La classe ?PredictedInteraction possède les attributs Protein1 et Protein2 pour désigner les deux protéines dont l'interaction est prédite. De plus, cette classe possède l'attribut Valid pour indiquer que cette prédiction est confirmée par l'interaction réelle fournie. Enfin, l'attribut Prediction précise la ou les règles prédictives qui ont produit cette prédiction.

Cette représentation des règles prédictives dans InterDB permet de profiter du langage de requêtes du système de gestion de bases de données acedb [Acedb] pour les étudier et les utiliser. En particulier, il est possible d'appliquer une classe de règles prédictives choisie à une ou plusieurs protéines d'intérêt, pour prédire des interactions entre ces protéines et d'autres protéines contenues dans InterDB.

5.4.b Application de règles prédictives

Il est nécessaire de développer un algorithme performant pour appliquer une classe de règles prédictives à une protéine d'intérêt. L'algorithme conçu recherche d'abord l'ensemble des règles valides de la classe choisie qui sont applicables à la protéine, c'est-à-dire les règles telles que tous les descripteurs correspondant à la protéine 1, dans la règle, sont des descripteurs de la protéine d'intérêt. L'algorithme identifie ensuite, pour chaque règle trouvée, l'ensemble des protéines de InterDB qui contiennent les descripteurs correspondant à la protéine 2 dans la règle, et qui proviennent du même organisme que la protéine d'intérêt.

L'étape cruciale de cet algorithme est la recherche de règles applicables à la protéine d'intérêt. En effet, une recherche naïve consisterait à produire tous les sous-ensembles de l'ensemble des descripteurs de la protéine d'intérêt, et pour chacun de ces sous-ensembles, à effectuer une requête dans InterDB afin de rechercher les règles prédictives pour lesquelles les descripteurs correspondant à la protéine 1 sont exactement ceux du sous-ensemble généré. Pour des protéines d'intérêt comportant un grand nombre de descripteurs, cet algorithme naïf conduit à effectuer beaucoup trop de requêtes dans InterDB. Par exemple, pour une protéine comportant 10 descripteurs il faudrait effectuer 2^{10} requêtes. Ceci n'est

pas souhaitable dans le cadre d'un programme qui doit pouvoir s'appliquer à des centaines de protéines d'intérêt et avec des dizaines de jeux de valeurs des paramètres de post-traitement.

Nous avons donc développé un algorithme efficace pour rechercher les règles applicables à une protéine d'intérêt. Cet algorithme utilise de manière complémentaire deux stratégies. Pour obtenir des règles comportant peu de descripteurs décrivant la protéine 1, la première stratégie, plus efficace dans ce cas, est appliquée. Par contre pour obtenir des règles comportant de nombreux descripteurs de la protéine 1, la seconde stratégie est plus avantageuse et est donc employée. L'algorithme est le suivant.

Soit SdS un entier positif. SdS est le seuil de stratégie, et détermine la stratégie à appliquer à chaque itération, comme suit. Tous les sous-ensembles de l'ensemble des descripteurs de la protéine d'intérêt comportant au maximum SdS descripteurs sont produits.

Pour les sous-ensembles comportant moins de SdS descripteurs, nous utilisons des requêtes "exactes", c'est-à-dire qu'elles imposent que les règles obtenues contiennent les descripteurs souhaités et aucun autre (pour la protéine 1). Les requêtes sont mises en œuvre facilement grâce à l'utilisation du COUNT du langage de requêtes de acedb [Acedb]. Par construction, les règles ainsi obtenues sont toujours applicables.

Par contre, pour les sous-ensembles de taille SdS, nous produisons des requêtes imposant aux règles obtenues de contenir au moins les descripteurs du sous-ensemble. Dans ce cas, nous vérifions ensuite que les règles potentiellement applicables obtenues sont effectivement applicables, c'est-à-dire que tous les descripteurs (correspondant à la protéine 1) de ces règles sont bien des descripteurs de la protéine d'intérêt.

Notons que nous obtenons bien en finale toutes les règles applicables à la protéine d'intérêt.

Exemple :

Supposons que l'on recherche les interacteurs potentiels d'une protéine décrite par les descripteurs D1, D2, D3 et D4. En fixant le seuil de stratégie SdS à 3, l'algorithme cherche les règles dans lesquelles la protéine 1 est décrite par :

D_i et aucun autre descripteur (avec $i \in \{1..4\}$);

D_i et D_j, et aucun autre descripteur (avec $i,j \in \{1..4\}$, $i \neq j$);

D_i, D_j et D_k, et éventuellement d'autres descripteurs (avec $i, j, k \in \{1..4\}$, $i \neq j \neq k \neq i$). De plus, dans ce cas, on vérifie que dans toute règle obtenue, la protéine 1 n'est caractérisée que par des descripteurs D_i avec $i \in \{1..4\}$.

Revenons sur la valeur du seuil de stratégie SdS. (SdS – 1) est en fait la valeur maximale du nombre de descripteurs à utiliser dans des requêtes exactes. Cette valeur a été fixée à 3 après expérimentations, de manière à optimiser les temps de calcul pour la recherche des interacteurs potentiels des protéines de notre jeu de validation (voir 5.5.b). Le programme Perl complet et commenté, nommé testpredictions.minex.updown.acestack.pl, est disponible en annexe 5.3.

5.5 Expérimentations et résultats

Dans ce chapitre, nous avons présenté précédemment le système d'extraction de connaissances que nous avons développé pour prédire des interactions entre protéines. Ce système comporte quatre composants dont les objectifs respectifs sont la modélisation des données, la recherche d'ensembles fréquents de descripteurs, l'extraction des ensembles fréquents significatifs, et l'exploitation des règles prédictives correspondantes. Le troisième composant, qui concerne l'extraction des ensembles significatifs, constitue l'étape cruciale du système et détermine en grande partie la qualité des prédictions qui peuvent être obtenues. Rappelons que ce composant met en œuvre quatre filtres de post-traitement, paramétrés par trois paramètres. A chaque jeu de valeurs de ces paramètres correspond une classe de règles prédictives.

Dans la suite, nous présentons d'abord les expérimentations menées pour produire les règles prédictives. Nous exposons ensuite les résultats obtenus au cours d'une campagne de tests concernant la prédiction d'interactions entre protéines du protéasome de *C. elegans*.

5.5.a Production de règles prédictives

On sait que les effets des quatre filtres de post-traitements conçus sont modulables par trois paramètres (voir 5.3). Afin d'étudier les effets de ces paramètres, nous avons procédé comme suit. Nous avons d'abord étudié et déterminé un ensemble de valeurs raisonnables pour chaque paramètre. Ensuite, pour chaque jeu de valeurs, nous avons appliqué les filtres et produit une classe de règles prédictives.

Dans cette partie, nous discutons d'abord du choix des valeurs des paramètres de post-traitement. Nous présentons ensuite les classes de règles prédictives obtenues. Nous exposons finalement les résultats en ce qui concerne les performances des filtres mis en œuvre.

5.5.a.1 Valeurs des paramètres de post-traitement

Le premier filtre (voir 5.3.a.1) écarte les ensembles fréquents qui contiennent un "mauvais" descripteur, c'est-à-dire un descripteur sans rapport avec les interactions entre protéines. Concrètement, cette notion doit être précisée par l'utilisateur, qui doit fournir la liste des descripteurs indésirables. Cette liste constitue le premier paramètre.

Nous avons observé que la majorité des "mauvais" descripteurs sont des mots-clés SwissProt. En première approximation, nous avons donc choisi de désigner comme "mauvais" certains mots-clés, selon le critère suivant : tout mot-clé qui caractérise plus de N protéines, parmi les protéines en interaction de InterDB, est "mauvais". N prend dans nos expérimentations les valeurs 0 (aucun mot-clé n'est utilisé), 50 (23 mots-clés sont écartés), et 100 (15 mots-clés écartés).

Exemple :

L'ensemble des mots-clés qui caractérisent plus de 100 protéines en interaction dans InterDB, et qui sont donc écartés quel que soit le seuil choisi, comprend par exemple *3D-structure*, *DNA-binding*, *Hypothetical_protein*, *Repeat*, ou encore *Multigene_family*. Ces mots-clés ne semblent pas présenter d'intérêt pour prédire des interactions entre protéines. Parmi les mots-clés qui sont également écartés au seuil 50, on trouve par exemple *Activator* et *Transport*, qui ne sont pas dénués d'intérêt mais risquent d'être trop généraux pour être exploitables.

Le second filtre (voir 5.3.a.2), qui écarte les ensembles ne concernant qu'une seule protéine, n'est pas paramétré.

Le troisième filtre (voir 5.3.a.3) définit la notion de score d'un ensemble fréquent, et élimine les ensembles dont le score est inférieur à une valeur qui constitue le troisième paramètre.

Dans nos expérimentations, nous avons choisi comme seuil de score les valeurs 2, 5, 10 et 100. La valeur 2 impose que pour tout ensemble fréquent valide, la fréquence observée soit supérieure au double de la fréquence attendue. Dans ce cas, certaines règles pourraient être retenues sans qu'elles soient réellement fondées. L'utilisation de l'autre valeur extrême, fixée à 100, produit des règles indubitablement statistiquement significatives, mais peut entraîner l'élimination de certaines règles correctes. En pratique, la meilleure valeur de

score-seuil à utiliser dépend des objectifs de l'utilisateur mais se situe probablement entre les deux valeurs extrêmes 2 et 100. Si l'utilisateur souhaite limiter la production de faux positifs il doit choisir un seuil élevé. Si au contraire l'utilisateur souhaite minimiser la production de faux négatifs, il doit plutôt opter pour un seuil relativement faible.

Le quatrième filtre (voir 5.3.a.4) élimine les ensembles fréquents qui sont sous-ensembles d'un ensemble fréquent, sous réserve que le rapport entre le score du sous-ensemble et le score du sur-ensemble soit inférieur à un seuil fourni en paramètre. Plus ce seuil est faible, plus les règles spécifiques sont favorisées par rapport aux règles générales.

Les valeurs retenues pour ce paramètre sont 1, 2, 3 et infini. En choisissant un seuil égal à 1, le filtre élimine tous les ensembles fréquents pour lesquels il existe un sur-ensemble fréquent de score supérieur. Cette condition sur les scores étant peu exigeante, les règles obtenues sont relativement spécifiques. A l'autre extrémité du spectre, une valeur infinie pour ce paramètre signifie que le quatrième filtre n'élimine aucun ensemble fréquent. Dans ce cas, la classe de règles prédictives comporte des règles générales, ce qui augmente le nombre d'interactions qui seront prédites mais diminue la fiabilité de ces prédictions. Ainsi, l'utilisateur a encore la possibilité de limiter plutôt la production de faux négatifs ou de faux positifs, selon la valeur choisie.

5.5.a.2 Analyse des classes de règles prédictives obtenues

Etant donné les domaines de valeurs choisis pour chaque paramètre de post-traitement, nous avons produit au total 48 classes de règles prédictives. Ces classes comportent entre 317 et 6123 règles, selon les valeurs des paramètres. Remarquons que pour toute règle asymétrique d'une classe, il existe dans la même classe une règle duale, obtenue en inversant dans la règle les descripteurs des deux protéines.

La classe la plus restrictive, qui comporte 317 règles, est produite en choisissant de ne retenir aucun mot-clé SwissProt, en fixant le seuil de score à 100 ce qui signifie que la fréquence observée des ensembles fréquents doit être supérieure à 100 fois leur fréquence attendue, et en fixant le troisième paramètre à 1 ce qui a pour effet d'écarter tout ensemble fréquent qui possède un sur-ensemble

fréquent de score supérieur. Cette classe contient les règles les plus sûres.

La classe la plus large comportant 6123 règles est produite en éliminant les mots-clés qui décrivent plus de 100 protéines en interaction dans InterDB, en choisissant 2 comme seuil de score et en fixant le paramètre du quatrième filtre à la valeur infinie, ce qui revient à passer outre ce filtre. Cette classe contient donc des règles générales comportant peu de descripteurs, bien que les règles plus spécifiques y apparaissent aussi. De plus, certaines règles de cette classe peuvent être discutablement significatives, étant donné le faible seuil de score employé. Cette classe contient donc les règles les moins sûres, mais pourrait permettre de prédire correctement des interactions que les autres classes rateraient.

Les classes qui correspondent à des valeurs intermédiaires des paramètres de post-traitement comportent typiquement entre 1000 et 2000 règles. A priori, ce sont ces classes que l'on souhaiterait généralement utiliser en premier lieu. Les expérimentations menées sur les protéines du protéasome du *C. elegans* (voir 5.5.b) confirment le bien-fondé de cette remarque.

5.5.a.3 Performances des filtres mis en œuvre

Les quatre filtres de post-traitement sont mis en œuvre dans les deux programmes filtre.minex.pl et filtre.2.minex.iter.pl (voir 5.3.b). Les post-traitements sont appliqués au fichier résultant de l'application de *min-ex* avec un seuil de fréquence égal à 0.0004. Ce fichier comporte 125744 ensembles fréquents condensés. Toutes nos expérimentations ont été effectuées sur un ordinateur sous Linux comportant un processeur Pentium II cadencé à 400 MHz et 256 Mo de RAM. Les performances obtenues sont les suivantes.

Le programme filtre.minex.pl met en œuvre les trois premiers filtres de post-traitement (voir 5.3.b.1). Il accepte en paramètre une liste de "mauvais" descripteurs et un seuil de score. Nous avons utilisé trois valeurs différentes pour le premier paramètre et quatre valeurs pour le second. Nous pouvons donc évaluer les performances du programme filtre.minex.pl au cours des douze exécutions correspondantes : en moyenne, une exécution dure 40 secondes sur notre machine.

Le programme filtre.2.minex.iter.pl met en œuvre le quatrième filtre, et accepte un paramètre qui prend 4 valeurs dans nos expérimentations. Comme ce filtre est appliqué à chaque fichier produit par filtre.minex.pl, nous pouvons donc

observer ses performances au cours de quarante-huit exécutions différentes : en moyenne, une exécution prend moins de 4 secondes sur notre machine.

Ces temps de calcul sont très raisonnables, et permettent de produire une classe de règles prédictives en moins d'une minute. Ainsi, il est tout à fait envisageable d'utiliser un jeu de valeurs spécifié par un collègue biologiste pour produire une nouvelle classe de règles prédictives, si les valeurs actuellement utilisées ne lui conviennent pas ou ne correspondent pas à ses besoins.

5.5.b Application aux protéines du protéasome de *C. elegans* : validation, résultats

Afin de mettre notre système de prédiction d'interactions à l'épreuve dans des conditions réelles, il est nécessaire de disposer de données d'interactions entre protéines qui soient sûres, c'est-à-dire expérimentalement déterminées, et qui n'aient pas été utilisées au cours de l'apprentissage. Fort à propos, nous avons participé récemment avec Anne Davy du CRBM de Montpellier et l'équipe de Marc Vidal au DFCI de Boston à un projet de cartographie entre protéines du protéasome de *C. elegans* [Dav01]. Par le biais de cette collaboration, nous disposons d'un jeu de données fraîches comportant 113 interactions mettant en cause 90 protéines. Aucune de ces interactions n'a été utilisée pour l'apprentissage. Elles constituent donc un très bon ensemble de validation.

Dans la suite, nous exposons tout d'abord la méthode employée pour exploiter cet ensemble de validation afin d'évaluer la qualité de notre système de prédiction d'interactions. Nous présentons ensuite les principaux résultats obtenus. Nous indiquons finalement les performances du programme de prédiction d'interactions mis en œuvre (voir 5.4.b).

5.5.b.1 Objectif et méthode

Rappelons que l'objectif final de notre système de prédiction d'interactions entre protéines est de guider les expériences de double hybride réalisées au DFCI dans l'équipe de Marc Vidal, afin d'accélérer la découverte d'interactions. En effet, bien que la détermination de l'interactome complet de *C. elegans* soit l'objectif à

long terme de cette équipe, la méthode employée consiste à produire dans un premier temps plusieurs cartes d'interactions de taille moyenne. Typiquement, un chercheur sélectionne 20 à 50 protéines impliquées dans un processus biologique donné, et recherche les interacteurs de ces protéines par criblages double hybride. L'idée est d'exploiter notre système prédictif pour proposer un ensemble d'interacteurs potentiels des protéines d'intérêt. Le collègue biologiste peut alors réaliser une expérience de double hybride par la méthode matricielle, peu coûteuse en temps, et ainsi tester les interactions prédites. De cette manière, une partie des interactions concernant les protéines d'intérêt peut être très rapidement découverte.

Par ailleurs, comme il est dit plus haut, nous disposons de 113 nouvelles interactions concernant 90 protéines [Dav01]. Ces interactions ont été découvertes par criblages double hybride (voir 3.2), et importées dans InterDB (voir chapitre 4). 23 protéines ont été utilisées en tant qu'appât. Au total dans ces 113 interactions, on trouve 80 protéines qui interagissent avec les appâts, dites interacteurs, dont 13 sont également des appâts. Remarquons cependant que 20 protéines parmi ces 80 interacteurs ne sont décrites par aucun descripteur dans InterDB. Plus précisément, ces protéines n'ont été introduites que récemment dans TrEMBL [Bai99] et ne disposent pas encore dans cette base de données d'annotation par mots-clés ou localisations cellulaires, ni d'affectation de domaine ou famille Interpro [Apw01]. Il s'ensuit que parmi les 113 interactions, au plus 86 pourraient être éventuellement prédites par notre système, dans la mesure où les (113 - 86 =) 27 interactions restantes mettent en cause au moins une de ces protéines qui ne comportent aucun descripteur. Ces 86 interactions, qui concernent 23 protéines appât, constituent donc notre jeu de validation.

Notre objectif est ici d'évaluer la qualité de notre système de prédiction d'interactions entre protéines, et d'identifier les classes de règles prédictives qui fournissent les meilleurs résultats. L'idée est de simuler une utilisation du système telle que celle décrite ci-dessus, puis d'évaluer la qualité des prédictions obtenues en les comparant aux interactions déterminées expérimentalement. Il s'agit donc de prédire les interacteurs potentiels des 23 protéines appât de notre jeu de validation.

La qualité des prédictions obtenues peut être évaluée selon deux critères.

D'une part, on peut considérer le pourcentage des interactions déterminées

expérimentalement qui sont prédites par le système. Ce taux représente la couverture du système prédictif, et sera désigné "taux de couverture" dans la suite. Plus le taux de couverture est important, plus le système est performant. En effet, les interactions correctement prédites sont celles dont l'utilisation de notre système prédictif aurait accéléré la découverte.

D'autre part, il est intéressant de connaître le pourcentage d'interactions prédites qui sont correctes, c'est-à-dire confirmées expérimentalement. Ce taux représente la spécificité du système prédictif. Il est donc nommé "taux de spécificité" dans la suite. Ce taux de spécificité ne peut pas être très élevé, en raison de la nature de notre système. En effet, la modélisation choisie pour représenter les protéines dans InterDB fait abstraction de la séquence protéique. Or, la spécificité des interactions entre protéines se trouve précisément dans cette séquence. Cependant, un faible taux de spécificité peut correspondre à un ensemble de prédictions tout à fait exploitable. En effet, les techniques employées au DFCI permettent de tester quelques milliers de couples de protéines (par exemple 5000) assez rapidement. Avec un taux de spécificité de l'ordre du pourcent, un tel travail pourrait révéler plusieurs dizaines d'interactions (en l'occurrence dans notre exemple, environ $5000/100 = 50$), ce qui serait très appréciable.

Il reste qu'afin d'évaluer l'intérêt de notre système, le taux de spécificité obtenu doit être comparé au taux équivalent d'un système, dit "modèle aléatoire", qui prédirait des interacteurs potentiels par tirage aléatoire. Ce taux équivalent peut être calculé comme suit.

D'une part, il apparaît raisonnable d'estimer qu'une protéine interagit en moyenne avec cinq autres protéines. D'autre part, le protéome de *C. elegans* comporte environ 20000 protéines. Il s'ensuit qu'en choisissant une paire de protéines au hasard, il y a grossièrement 5 chances sur 20000 que les deux protéines interagissent. Le taux de spécificité d'un modèle aléatoire est donc de l'ordre de 5/20000, soit 0.025%.

Il apparaît finalement qu'un taux de spécificité de l'ordre de 2.5% constitue un gain d'un facteur 100 par rapport au tirage aléatoire. Un tel gain est appréciable. S'il est atteint notre système prédictif s'avèrerait très utile dans le cadre du projet de cartographie des interactions entre protéines en cours au DFCI de Boston.

5.5.b.2 Résultats

Nous avons appliqué 36 des 48 classes de règles prédictives produites (voir 5.5.a) aux 23 protéines appât de notre jeu de validation : les classes de règles produites en fixant la valeur du paramètre du quatrième filtre à 2 ont été écartées, car elles se sont révélées très similaires aux classes où ce paramètre vaut 1 ou 3. Pour chaque classe, nous obtenons ainsi un ensemble de paires de protéines, dont l'une est un des 23 appâts et dont l'interaction est prédite par une ou plusieurs règles de la classe. Par comparaison de ces 36 ensembles de prédictions d'interactions avec l'ensemble des interactions réelles du jeu de validation, on associe alors à chaque classe de règles un taux de couverture et un taux de spécificité.

Afin de présenter les résultats, il est nécessaire de pouvoir désigner les classes de règles prédictives. Dans la suite, nous représentons chaque classe de règles sous la forme d'un triplet de la forme : (seuil d'élimination des mots-clés, seuil de score des ensembles fréquents, seuil d'élimination des sous-ensembles d'ensembles fréquents). Un tel triplet comprend les valeurs des paramètres des filtres 1, 3 et 4 respectivement, qui ont permis de produire la classe de règles considérée.

Exemple :

La classe représentée par (100, 10, 3) est la classe de règles produite en écartant les mots-clés qui décrivent plus de 100 protéines en interaction dans InterDB, en éliminant les ensembles fréquents de score inférieur à 10, et en éliminant les ensembles fréquents s'ils sont sous-ensembles d'un autre ensemble fréquent dont le score est trois fois supérieur aux leurs.

Les résultats obtenus peuvent être analysés de la façon suivante.

- Taux de couverture :

Selon la classe de règles, nous avons pu prédire correctement 2, 5, 10, 15, 17 ou 18 interactions parmi les 86 interactions du jeu de validation. Les taux de couverture obtenus varient donc de 2.3% à 20.9%.

- Taux maximal de spécificité :

A chaque taux de couverture TC, nous associons un taux maximal de

spécificité, qui permet une bonne appréhension globale des résultats. Ce taux maximal de spécificité est défini comme suit. Considérons l'ensemble des classes de règles dont le taux de couverture est égal à TC. Un taux de spécificité caractérise chacune de ces classes. Le taux maximal de spécificité associé à TC est le plus grand taux de spécificité qui caractérise une des classes de règles dont le taux de couverture est égal à TC. Ce taux permet donc d'évaluer globalement la qualité du système prédictif dans le meilleur des cas.

Les taux maximaux de spécificité varient entre 0.6% et 5.9%. Ces résultats sont récapitulés dans la figure 5.2.

Nb de prédictions correctes	Taux de couverture	Taux maximal de spécificité	Classes de règles concernées
2	2.3%	5.9%	(0, 100, *)
5	5.8%	2.0%	(0, 10, *)
10	11.6%	4.0%	(50, 100, *) et (100, 100, *)
15	17.4%	3.2%	(50, 10, 1) et (100, 10, 1)
17	19.8%	3.0%	(50, 10, 3) et (100, 10, 3)
18	20.9%	0.6%	(50, 2, infini)

Figure 5.2 : Taux de couverture, taux maximaux de spécificité et classes de règles correspondantes.

Pour chaque valeur du nombre de prédictions correctes obtenue, le taux de couverture correspondant et le taux maximal de spécificité associé sont indiqués. Les classes de règles qui possèdent le taux maximal de spécificité sont précisés, sous la forme d'un triplet (seuil d'élimination des mots-clés, seuil de score des ensembles fréquents, seuil d'élimination des sous-ensembles d'ensembles fréquents). Dans ce tableau, une astérisque est employée pour désigner toutes les valeurs possibles d'un paramètre.

Au vu des expérimentations menées à l'aide de notre jeu de validation, nous pouvons identifier les classes de règles les plus prometteuses. En pratique, nous estimons que le taux de couverture doit être supérieur à 10% pour que l'exploitation des prédictions procure un gain réellement appréciable. Ainsi, en fonction des objectifs de l'utilisateur biologiste, certaines classes de règles sont à employer en priorité.

D'un côté, si l'utilisateur souhaite identifier rapidement un faible nombre d'interactions, il doit privilégier les classes de règles à forte spécificité et faible couverture. Dans ce cas, il apparaît que les règles (50, 100, *) et (100, 100, *) sont les plus indiquées. Ces règles correspondent au seuil de score maximal utilisé, à savoir 100. A ce seuil, le premier filtre ne doit pas écarter tous les mots-clés, sous peine de voir le taux de couverture chuter à 2.3%. Le quatrième filtre ne semble pas influencer le contenu des classes lorsque le seuil de score est aussi important. Ce constat s'explique par le fait que les scores des règles trop générales, qui sont la cible du quatrième filtre, n'atteignent pas un tel seuil. En fait, le quatrième filtre est plutôt conçu pour trier les règles de score intermédiaire. Ainsi son effet est apparent si l'on considère les règles (*, 10, *) qui correspondent à un seuil de score de 10.

D'un autre côté, l'utilisateur peut vouloir identifier un nombre plus important d'interactions, quitte à tester proportionnellement plus de couples de protéines. Dans ce cas il faut employer des classes à forte couverture, sans pour autant laisser trop choir le taux de spécificité. Il apparaît que les classes les plus indiquées sont (50, 10, 1), (100, 10, 1), (50, 10, 3), et (100, 10, 3). On peut remarquer que ces classes utilisent toutes un seuil de score de 10, et au moins quelques mots-clés. Le quatrième filtre influence ici légèrement la couverture et la spécificité selon la valeur de son paramètre, mais joue en tous cas un rôle positif. En effet, s'il est ignoré en spécifiant la valeur infinie pour son paramètre, le taux de spécificité tombe à 2.6% sans gain de couverture (voir les données disponibles en annexe 5.4).

A l'issue de ces expérimentations menées à l'aide de notre jeu de validation, on peut tirer les conclusions suivantes.

Tout d'abord, le système de prédiction d'interactions que nous avons développé est tout à fait exploitable. Il permet en effet d'atteindre des taux de couverture très respectables de l'ordre de 15%, tout en conservant une spécificité

supérieure à 3%. Rappelons qu'une spécificité de 2.5% représente un gain d'un facteur 100 par rapport au modèle aléatoire, qui constitue l'alternative naïve naturelle dans le cadre d'un projet de cartographie à grande échelle d'interactions entre protéines.

On constate ensuite qu'il n'apparaît pas utile de choisir un seuil de score inférieur à 10, car la spécificité du système prédictif est alors trop faible. De plus, il est important de conserver une partie des mots-clés SwissProt, sans quoi la couverture du système prédictif se dégrade. Enfin, l'élimination des règles trop générales par le quatrième filtre se révèle efficace si le seuil de score reste modéré.

Un résumé complet des résultats, comprenant le nombre d'interactions prédites ainsi que le nombre de prédictions correctes pour chaque classe de règles, est disponible en annexe 5.4.

5.5.b.3 Performances du programme de prédiction d'interactions

Au cours de cette campagne d'expérimentations, nous avons pu évaluer les performances du programme *testpredictions.minex.updown.acestack.pl*, qui applique une classe de règles prédictives à un ensemble de protéines d'intérêt et permet ainsi de prédire des interactions entre protéines (voir 5.4.b). Toutes nos expérimentations ont été effectuées sur un ordinateur sous Linux comportant un processeur Pentium II cadencé à 400 MHz et 256 Mo de RAM.

Rappelons que ce programme recherche les règles prédictives applicables, puis pour chaque règle applicable il identifie les protéines qui interagissent avec la protéine d'intérêt selon cette règle. Les données à examiner, stockées dans la base InterDB, sont complexes et nombreuses : InterDB contient plus de 463000 protéines et plus de 114000 règles prédictives. Afin de permettre une utilisation effective du système prédictif dans des durées raisonnables, nous avons apporté diverses optimisations au programme (voir 5.4.b).

En particulier, nous avons conçu un algorithme original qui exploite deux stratégies complémentaires : selon le nombre de descripteurs de la protéine d'intérêt considérés dans une itération donnée, l'une ou l'autre des stratégies est employée. Le seuil utilisé pour choisir la stratégie à appliquer est un paramètre de cet algorithme. C'est à la suite d'une série d'expérimentations que nous avons fixé

la valeur de ce seuil de stratégie à 3. Ces expérimentations sont les suivantes.

Nous avons sélectionné la classe de règles prédictives (100, 10, 1), qui apparaît comme une classe représentative selon les paramètres des post-traitements, et utilisé l'algorithme pour appliquer cette classe aux 23 protéines appât de notre jeu de validation, en fixant successivement le seuil de stratégie aux valeurs 2, 3, 4, et 5. Les interactions prédites sont bien entendu identiques, puisque le seuil de stratégie n'affecte pas la complétion de l'algorithme. Cependant, les temps de calcul observés sont variables, comme le montre la figure 5.3. On constate bien que les performances du programme sont maximales lorsque le seuil de stratégie prend la valeur 3.

Seuil de stratégie	Temps de calcul (h:mm:ss)
1	1:20:49
2	1:18:24
3	1:16:50
4	1:16:53
5	1:16:57
20	1:22:26

Figure 5.3 : Temps de calcul pour prédire les interacteurs des 23 protéines appât de notre jeu de validation, en fonction de la valeur choisie pour le seuil de stratégie.
La valeur 1 correspond à l'utilisation exclusive de la seconde stratégie. Réciproquement, la valeur 20 correspond à l'utilisation exclusive de la première stratégie, dans la mesure où aucune protéine d'InterDB n'est décrite par plus de 20 descripteurs.

Ce paramètre étant fixé à 3, nous avons observé le temps moyen d'exécution du programme, pour les 36 exécutions successives correspondant aux 36 classes de règles prédictives appliquées aux 23 protéines appât de notre jeu de validation. En moyenne, appliquer une classe de règles prédictives à ces 23 protéines prend 1 heure et 16 minutes sur notre ordinateur. Cela correspond à 3

minutes et 18 secondes en moyenne pour rechercher les interacteurs potentiels d'une protéine d'intérêt. De plus, on constate que le temps de calcul varie relativement peu selon les classes de règles appliquées.

Ces temps de calcul sont plutôt élevés pour un programme destiné à prédire les interacteurs de dizaines de protéines d'intérêt, mais ils restent raisonnables. Il n'est pas prohibitif d'attendre quelques heures pour obtenir les prédictions d'interactions correspondant à plusieurs dizaines de protéines, étant donné que ces prédictions peuvent permettre d'obtenir rapidement des résultats qui prendraient normalement des semaines, voire des mois de travail.

6. Conclusion et perspectives

Nous présentons dans ce chapitre les conclusions que nous tirons de ce travail de thèse, et les perspectives qui s'offrent à nous pour la poursuite des recherches.

6.1 Bilan

Au cours de ces travaux de thèse, j'ai effectué deux longs séjours à Boston au sein de l'équipe du professeur Marc Vidal, au Massachusetts General Hospital Cancer Center puis au Dana Farber Cancer Institute (DFCI). Ces séjours m'ont été extrêmement bénéfiques sur le plan scientifique, puisqu'ils m'ont permis de m'initier à la biologie en général et aux sciences du génome en particulier.

J'ai ainsi pu participer aux deux projets de génomique fonctionnelle qui constituent le cœur des recherches menées dans l'équipe de Marc Vidal.

Il s'agit d'une part du projet de clonage de l'ORFéome, ou ensemble des zones codantes, de *C. elegans* [Reb01]. Ce projet repose sur une amplification par PCR de zones codantes prédites, suivie du clonage des produits obtenus dans le vecteur navette du système Gateway [Wal00b] de clonage par recombinaison. A cette fin, j'ai développé un programme informatique pour automatiser l'étape de conception des amorces de PCR. Dans le cadre du système Gateway, il est possible de transférer les gènes clonés vers un grand nombre de vecteurs d'expression. Au DFCI, nous utilisons en particulier comme vecteurs de destination deux vecteurs spécifiques du système double hybride, dans le cadre du projet de cartographie des interactions entre protéines. Mais plus généralement, notre projet de clonage de l'ORFéome doit aboutir à la constitution d'une ressource qui pourrait s'avérer très utile pour d'autres projets de génomique fonctionnelle chez *C. elegans*.

D'autre part, le second projet concerne la cartographie des interactions protéine-protéine chez *C. elegans* ([Wal00], [Dav01]). Le système double hybride apparaît comme une bonne solution pour l'identification des interactions entre protéines dans un cadre de génomique fonctionnelle. Le projet pilote de

cartographie des interactions entre protéines impliquées dans le développement de la vulve chez *C. elegans* [Wal00] nous a conduit à développer des protocoles et une plate-forme informatique, nommée WISTdb, susceptibles de s'appliquer à l'échelle d'un génome. La plate-forme WISTdb, que j'ai conçue et mise en œuvre, remplit un grand nombre de fonctions essentielles au bon déroulement du projet de cartographie des interactions chez *C. elegans*, depuis la production et le stockage des données expérimentales jusqu'au traitement et à l'analyse des résultats.

Une seconde étape du travail a consisté à construire InterDB [Thi01a], une base de données multi-organismes d'interactions protéine-protéine orientée-prédiction. Notre démarche a été la suivante.

Nous avons sélectionné deux bases de données de protéines, SwissProt et TrEMBL, qui sont complémentaires et bénéficient, de l'avis des experts, d'annotations de bonne qualité. Nous avons identifié trois sources de données d'interactions protéine-protéine déterminées expérimentalement : WISTdb, YPD et DIP. Nous avons recherché des sources de caractérisation fonctionnelle ou structurale des protéines, susceptibles d'expliquer les interactions observées et donc d'en prédire de nouvelles. Dans cette optique, nous nous sommes concentrés sur les descripteurs disponibles dans les bases de motifs, domaines et familles protéiques, telles que ProSite et Pfam, et avons migré dès que cela a été possible vers la base fédératrice Interpro. Nous exploitons également certaines annotations provenant de SwissProt et TrEMBL : les mots-clés, et les localisations cellulaires.

En intégrant toutes ces sources de données, nous avons construit une base de données multi-organismes d'interactions entre protéines orientée-prédiction, dénommée InterDB. Cette base originale contient actuellement 2464 interactions concernant 2032 protéines, provenant principalement de la levure *S. cerevisiae*, de l'Homme et du nématode *C. elegans*.

Pour assurer la construction d'InterDB, nous avons développé un système logiciel composé d'un ensemble d'interpréteurs pour les bases d'origine. Ces interpréteurs produisent une représentation des données au format *.ace*, conformément au schéma choisi pour InterDB. Notre système permet de mettre facilement InterDB à jour lorsque de nouvelles données deviennent disponibles dans les bases source.

Ce travail était relativement technique mais nécessaire. Je l'ai accompli

avec plaisir et j'en ai retiré une certaine maîtrise du langage Perl et de l'interface AcePerl, qui m'ont servi par la suite et me serviront certainement dans le futur.

Nous avons finalement développé un système de prédiction d'interactions protéine-protéine en exploitant la base de données InterDB. Il s'agit d'un système d'extraction de connaissances, basé sur l'algorithme de Datamining *min-ex* [Bou00]. Nous avons eu accès à une implémentation efficace de cet algorithme à travers une collaboration avec l'équipe de Jean-François Boulicaut, du LISI de l'INSA-Lyon, au sein de laquelle cet algorithme a été développé.

Notre système de prédiction d'interactions comporte quatre composants dont les objectifs respectifs sont la modélisation des données, la recherche d'ensembles fréquents de descripteurs, l'extraction des ensembles fréquents significatifs, et l'exploitation des règles prédictives correspondantes. Le troisième composant, qui concerne l'extraction des ensembles significatifs, constitue l'étape cruciale du système et détermine en grande partie la qualité des prédictions qui peuvent être obtenues. Ce composant met en œuvre quatre filtres de post-traitement, paramétrés par trois paramètres. A chaque jeu de valeurs de ces paramètres correspond une classe de règles prédictives.

Nous avons pu évaluer la qualité de notre système prédictif au cours d'une campagne d'expérimentations, conduite à l'aide d'un jeu de validation comportant 113 interactions entre protéines du protéasome de *C. elegans*. Ces données proviennent d'une récente collaboration [Dav01] avec Anne Davy, du Centre de Recherches en Biologie Moléculaire de Montpellier, et avec l'équipe de Marc Vidal.

Rappelons que notre objectif est de guider les expériences de double hybride réalisées au DFCI dans l'équipe de Marc Vidal, afin d'accélérer la découverte d'interactions. Partant d'un ensemble de protéines auxquelles s'intéresse un membre de cette équipe, l'idée est d'utiliser notre système prédictif pour proposer un ensemble d'interacteurs potentiels des protéines d'intérêt. Le collègue biologiste peut alors réaliser des expériences de double hybride selon la méthode matricielle, peu coûteuse en temps, pour tester les interactions prédites. De cette manière, une partie des interactions concernant les protéines d'intérêt peut être très rapidement découverte.

Les résultats obtenus nous paraissent satisfaisants bien que perfectibles, et susceptibles de répondre au moins partiellement à l'objectif fixé. En effet, en

choisissant convenablement la classe de règles prédictives à appliquer, nous parvenons à prédire près de 20% des interactions de notre jeu de validation, avec un taux de spécificité de l'ordre de 3%. Ainsi, en testant 560 couples de protéines par double hybride, nous aurions pu découvrir très rapidement un cinquième des interactions qui ont été révélées par criblage double hybride.

Pour conclure, rappelons que la thèse défendue est que les interactions entre protéines sont en partie explicables et prédictibles à partir des annotations disponibles dans les bases de données publiques, sous réserve que l'on dispose de suffisamment d'exemples de couples de protéines qui interagissent effectivement. A l'issue des travaux effectués, je pense pouvoir affirmer que cette thèse est bien valide.

Il est intéressant de remarquer qu'une publication très récente confirme également le bien-fondé de cette thèse [Spr01]. Cet article rend compte d'un travail dont l'objectif est assez proche du notre. Les auteurs ont construit une base de données sur le modèle de InterDB, bien que les seuls descripteurs utilisés soient les domaines et familles InterPro. Ils exploitent ensuite cette base de données pour construire une matrice d'affinité entre descripteurs, exactement comme nous l'avions précédemment décrit dans [Thi00]. Ils obtiennent ainsi des couples de descripteurs InterPro qui sont significativement liés. La méthode informatique et statistique, identique à [Thi00], est relativement rudimentaire comparée à notre méthode d'apprentissage basée sur l'extraction d'ensembles fréquents, mais elle se révèle efficace dans leurs expérimentations. L'approche permet d'obtenir des règles prédictives comportant un unique descripteur pour chaque protéine. Si l'on compare la forme des résultats obtenus, notre système de prédiction d'interactions entre protéines peut être perçu comme une généralisation du système décrit, puisqu'il produit des règles prédictives comportant un nombre quelconque de descripteurs pour chaque protéine. Néanmoins, conceptuellement l'idée est la même, et ce travail constitue bien une confirmation de la thèse que je défends.

6.2 Perspectives

Rappelons que ce travail de thèse comporte trois volets. Chacun de ces volets offre des perspectives à plus ou moins long terme. Elles sont présentées dans cette section.

D'une part, nous avons contribué aux projets de génomique fonctionnelle en cours dans l'équipe de Marc Vidal au DFCI, en particulier par le développement de la plate-forme informatique WISTdb. Cette plate-forme doit évoluer avec les techniques expérimentales qui sont employées au DFCI. De plus, la base de données qui est au cœur de cette plate-forme doit être maintenue à jour, afin de rester cohérente avec WormBase [Ste01], la base publique *C. elegans*.

D'autre part, nous avons développé InterDB, une base de données fédérative d'interactions entre protéines multi-organismes. Les perspectives d'évolution d'InterDB sont multiples. Au delà de la maintenance de la base qui nécessite une reconstruction périodique en fonction des mises à jour des bases d'origine, nous envisageons d'incorporer des interactions provenant de sources récentes et prometteuses, en particulier la base BIND [Bad01] et les travaux de Ito *et al* [Ito01]. Par ailleurs, nous souhaitons intégrer de nouveaux descripteurs, découlant par exemple des travaux du Gene Ontology Consortium [Ash00] qui propose une classification hiérarchique des gènes de la levure, de la souris, de l'Homme, et bientôt de *C. elegans*.

Enfin, nous avons conçu et mis en œuvre un système de prédiction d'interactions entre protéines, dont l'objectif est de guider les expérimentations de double hybride menées au DFCI afin d'accélérer la découverte d'interactions protéine-protéine. Nous avons conduit une campagne d'évaluation de ce système prédictif, qui confirme le bien-fondé de la méthode. L'étape suivante consiste à exploiter et valider cet outil en conditions réelles, dans le cadre du projet de cartographie systématique des interactions entre protéines. Ce faisant, nous espérons pouvoir perfectionner encore le système, en profitant de l'expérience et des connaissances de ses utilisateurs biologistes.

D'une manière plus générale, il est clair que les performances de notre

système prédictif dépendent fortement des connaissances biologiques, c'est-à-dire des interactions disponibles dans InterDB d'une part, et des annotations utilisées comme descripteurs des protéines d'autre part.

L'obtention de nouvelles interactions entre protéines ne peut qu'améliorer la qualité de notre système, sous réserve que celles-ci soient authentiques. Afin de garantir ce point, nous souhaitons poursuivre la politique suivie jusqu'à présent, qui consiste à n'intégrer dans InterDB que des interactions déterminées expérimentalement et provenant de sources fiables. Pour prendre en compte de telles interactions, il suffit de les intégrer dans InterDB, et de reconstruire notre système prédictif en exploitant le nouveau contenu de la base pour l'apprentissage.

Par contre, pour exploiter de nouveaux descripteurs, il peut être nécessaire de se tourner vers d'autres méthodes d'apprentissage. En effet, la méthode d'extraction de connaissances basée sur la recherche d'ensembles fréquents, que nous utilisons actuellement, traite des données plates, dépourvues de structure. Cette technique n'est donc pas très bien adaptée pour exploiter au mieux des descripteurs structurés, comme par exemple les annotations proposées par le Gene Ontology Consortium [Ash00]. Dans cette optique, d'autres méthodes d'apprentissage pourraient se révéler plus adéquates, par exemple le système WarmR [Deh99].

Bibliographie

(Nombre total de références bibliographiques : 110.)

Afin d'en faciliter l'utilisation, nous avons choisi de classer les références bibliographiques dans trois sections, en fonction de leur orientation thématique. Nous présentons d'abord les documents à caractère résolument biologique. Figurent ensuite les travaux concernant les bases de données biologiques, ainsi que les références en bio-informatique. Nous présentons finalement les références relevant exclusivement de l'informatique.

Biologie

[Ada00] Adams MD et al (2000) : The genome sequence of *Drosophila melanogaster*. *Science* 287, 2185-2195.

[Ahr97] Ahringer J (1997) : Turn to the worm. *Curr Opin Genet Dev* 7:410-415.

[Cel98] The *C. elegans* Sequencing Consortium (1998) : Genome sequence of the nematode *C. elegans* : a platform for investigating biology. *Science* 282, 2012-2018.

[Dav01] A Davy, P Bello, N Thierry-Mieg, P Vaglio, J Hitti, L Doucette-Stamm, D Thierry-Mieg, J Reboul, S Boulton, A Walhout, O Coux, M Vidal (2001) : A protein-protein interaction map of the *C. elegans* 26S proteasome. *EMBO Reports* 2(9): 821-828.

[Fie89] Fields, S. and Song, O. (1989) : A novel genetic system to detect protein-protein interactions. *Nature* 340, 245-246.

[Fra00] Fraser AG, Kamath RS, Zipperlen P, Martinez-Campos M, Sohrmann M, Ahringer J (2000) : Functional genomic analysis of *C. elegans* chromosome I by systematic RNA interference. *Nature* 408, 325-330.

[Gof96] Goffeau A, Barrell BG, Bussey H, Davis RW, Dujon B, Feldmann H, Galibert F, Hoheisel JD, Jacq C, Johnston M, Louis EJ, Mewes HW, Murakami Y, Philippsen P, Tettelin H, Oliver SG (1996) : Life with 6000 genes. *Science* 274(5287):546, 563-7.

[Gon00] Gonczy P *et al* (2000) : Functional genomic analysis of cell division in *C. elegans* using RNAi of genes on chromosome III. *Nature* 408, 331-336.

[Hil00] Hill A, Hunter C, Tsung B, Tucker-Kellogg G, Brown E (2000) : Genomic analysis of gene expression in *C. elegans*. *Science* 290, 809-812.

[Ito00] Ito T, Tashiro K, Muta S, Ozawa R, Chiba T, Nishizawa M, Yamamoto K, Kuhara S, Sakaki Y (2000) : Toward a protein-protein interaction map of the budding yeast : a comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *PNAS* 97(3), 1143-1147.

[Ito01] Ito T, Chiba T, Ozawa R, Yoshida M, Hattori M, Sakaki Y (2001) : A comprehensive two-hybrid analysis to explore the yeast protein interactome. *PNAS* 98(8), 4569-4574.

[Jia01] Jiang M, Ryu J, Kiraly M, Duke K, Reike V, Kim S (2001) : Genome-wide analysis of developmental and sex-regulated gene expression profiles in *Caenorhabditis elegans*. *PNAS* 98(1), 218-223.

[Jon96] Jones, S. And Thornton, J.M. (1996) : Principles of protein-protein interactions derived from structural studies. *PNAS* 93(1), 13-20.

[Kim] The Kim laboratory, http://cmgm.stanford.edu/~kimlab/.

[KOC] The *C. elegans* Gene Knockout Consortium, http://www.cigenomics.bc.ca/elegans/.

[Kuw97] Kuwabara PE (1997) : Worming your way through the genome. *Trends*

in genetics 13:455-460.

[Lan01] Lander ES *et al.* (2001) : Initial sequencing and analysis of the human genome. *Nature* 409(6822):860-921.

[Lec98] Lecrenier N., Foury F., Goffeau A. (1998) : Two-hybrid systematic screening of the yeast proteome. *BioEssays*, 20, 1-5.

[Mae01] Maeda I, Kohara Y, Yamamoto M and Sugimoto A (2001) : Large-scale analysis of gene function in *Caenorhabditis elegans* by high-throughput RNAi. *Current Biology* 11,171-176.

[Pia00] Piano F, Schetterdagger AJ, Mangone M, Stein L, Kemphues KJ (2000) : RNAi analysis of genes expressed in the ovary of *Caenorhabditis*. *Current Biology* 10, 1619-1622.

[Rai01] JC Rain, L Selig, H De Reuse, V Battaglia, C Reverdy, S Simon, G Lenzen, F Petel, J Wojcik, V Schachter, Y Chemama, A Labigne, P Legrain (2001) : The protein-protein interaction map of Helicobacter pylori. *Nature*, 409(6817):211-215.

[Reb01] J Reboul, P Vaglio, N Tzellas, N Thierry-Mieg, T Moore, C Jackson, T Shin-i, Y Kohara, D Thierry-Mieg, J Thierry-Mieg, H Lee, J Hitti, L Doucette-Stamm, J Hartley, G Temple, M Brasch, J Vandenhaute, P Lamesch, D Hill, M Vidal (2001) : Open-reading-frame sequence tags (OSTs) support the existence of at least 17300 genes in *C. elegans*. *Nature Genetics*, 27(3), 332-336.

[Rei00] Reinke V, Smith H, Nance J, Wang J, Van Doren C, Begley R, Jones S, Davis E, Scherer S, Ward S, Kim S (2000) : A global profile of Germline expression in *C. elegans*. *Molecular Cell* 6:605-616.

[Sal99] A. Sali (1999) : Functional links between proteins. *Nature* 402, 23-26.

[Sar97] C. Sardet, L. LeCam, E. Fabrizio, and M. Vidal (1997) : E2Fs and the

Retinoblastoma family. In *Progress in Gene Expression*, eds. M. Karin, M. Yaniv, and J. Ghysdael, Birkhauser publishing LTD, Basel, Switzerland.

[Ste01] Sterberg PW (2001) : Working in the post-genomic *C. elegans* world. *Cell* 105(2), 173-176.

[Thi01b] Thierry-Mieg D et J, Suzuki Y, Sugano S, Oishi K, Sano M, Nomoto H, Haga S, Nishizaka S, Hayashi H, Ohta F, Miura S, Uesugi H, Potdevin M, Thierry-Mieg Y, Simonyan V, Lowe A, Shin-i T, Kohara Y (2001) : The worm transcriptome project. *13th International C. elegans Meeting*, June 2001, University of California Los Angeles.

[Uet00] Uetz *et al* (2000) : A comprehensive analysis of protein-protein interactions in *Saccharomyces cerevisiae*, Nature, 403, 623-627.

[Ver95] Vernet T., Berti P.J., De Montigny C., Musil R., Tessier D.C., Ménard R., Magny M.C., Storer A.C., et Thomas D.Y. (1995) : Processing of the papain precursor : the ionization state of a conserved amino acid motif within the pro region participates in the regulation of intramolecular processing. *J. Biol. Chem.* 270, 10838-10846.

[Vid96a] Marc Vidal, Rainer K. Brachmann, Ali Fattaey, Ed Harlow, Jef D. Boeke (1996) : Reverse two-hybrid and one-hybrid systems to detect dissociation of protein-protein and DNA-protein interactions. *PNAS*, Vol 93, 10315-10320.

[Vid96b] Marc Vidal, Pascal Braun, Elbert Chen, Jef D. Boeke, Ed Harlow (1996) : Genetic characterization of a mammalian protein-protein interaction domain by using a yeast reverse two-hybrid system. *PNAS*, Vol 93, 10321-10326.

[Vid98] Vidal M, Endoh H, Thierry-Mieg N, Walhout M, Wong W : Description of a protein-protein interaction mapping project. 1998 East Coast *C. elegans* Meeting, June 1998, Boston University.

[Vid99] M. Vidal, P. Legrain (1999) : Yeast forward and reverse 'n'-hybrid

systems. *Nucleic Acids Research* 27(4), 919-929.

[Vid99b] Vidal M, Walhout M, Sordella R, Thierry-Mieg N, Brasch M, Temple G, Hartley J, Lorson M, van den Heuvel S, Endoh H: The *C. elegans* protein interaction mapping project : a test-case using proteins involved in vulval development. 12th Annual Meeting on Genome Mapping, Sequencing & Biology, May 1999, Cold Spring Harbour Laboratory.

[Wal98] A. Walhout, H. Endoh, N. Thierry-Mieg, W. Wong, M. Vidal (1998) : A model of elegance. *American Journal of Human Genetics* 63(4), 955-61.

[Wal00] A. Walhout, R. Sordella, X. Lu, J. Hartley, G. Temple, M. Brasch, N. Thierry-Mieg, M. Vidal (2000) : Protein interaction mapping in *C. elegans* using proteins involved in vulval development. *Science*, 287, 116-122.

[Wal00b] Walhout AJM et al (2000) : Gateway recombinational cloning: application to the cloning of large numbers of open reading frames, or ORFeomes. *Methods in Enzymology*, 328, 575-592.

[Wat94] Watson, Gilman, Witkowski, Zoller (1994) : ADN Recombinant, 2ème edition, DeBoeck Université, 1994.

[Whi96] White M. (1996) : The yeast two-hybrid system: Forward and reverse. *PNAS*, Vol 93, 10001-10003.

Bio-informatique, bases de données biologiques

[Acedb] The ACEDB documentation library, http://genome.cornell.edu/acedoc/. See also http://www.acedb.org.

[Acembly] The Acembly sequence assembly package, http://alpha.crbm.cnrs-mop.fr/acembly/.

[Ash00] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G (2000) : Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics* 25(1):25-29.

[Att97] Attwood, T.K., Beck, M.E., Bleasby, A.J., Degtyarenko, K., Michie, A.D. And Parry-Smith, D.J. (1997) : Novel developments with the PRINTS protein fingerprint database. *Nucleic Acids Research*, 25(1), 212-216.

[Att99] T.K. Attwood, D.R. Flower, A.P. Lewis, J.E. Mabey, S.R. Morgan, P. Scordis, J.N. Selley, W. Wright (1999) : PRINTS prepares for the new millenium. *Nucleic Acids Research*, 27(1), 220-225.

[Apw01] Apweiler R, Attwood TK, Bairoch A, Bateman A, Birney E, Biswas M, Bucher P, Cerutti L, Corpet F, Croning MDR, Durbin R, Falquet L, Fleischmann W, Gouzy J, Hermjakob H, Hulo N, Jonassen I, Kahn D, Kanapin A, Karavidopoulou Y, Lopez R, Marx B, Mulder NJ, Oinn TM, Pagni M, Servant F, Sigrist CJA, Zdobnov EM (2001) : The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Research*, 29(1), 37-40.

[Bab98] Olivier Baby (1998) : Non-deterministic, constraint-based parsing of human genes. PhD thesis, Brandeis University, Mass.

[Bad01] Bader GD, Donaldson I, Wolting C, Ouellette BF, Pawson T, Hogue CW (2001) : BIND--The Biomolecular Interaction Network Database. *Nucleic Acids Research* 29(1), 242-245.

[Bai99] Bairoch A. and Apweiler R. (1999) : The SWISS-PROT protein sequence data bank and its supplement TrEMBL. *Nucleic Acids Research* 27(1), 49-54.

[Bal98] P. Baldi, S. Brunak (1998) : Bioinformatics : the machine learning approach. MIT press, Cambridge, Mass.

[Bat99] A. Bateman, E. Birney, R. Durbin, S. Eddy, R.D. Finn, E.L. Sonnhammer (1999) : Pfam 3.1 : 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Research*, 27(1), 260-262.

[Bla99] Blaschke C, Andrade MA, Ouzounis C, Valencia A. (1999) : Automatic extraction of biological information from scientific text: protein-protein interactions. *Proceedings of ISMB99*, pp 60-67.

[Cor99] F. Corpet, J. Gouzy, D. Kahn (1999) : Recent improvements of the ProDom database of protein domain families. *Nucleic Acids Research*, 27(1), 263-267.

[Dur94] Durbin R. and Thierry-Mieg J. (1994) : The ACeDB genome database. In Suhai S. (ed.) : *Computational methods in genome research*, Plenum Press, New York.

[Dur98] R. Durbin, S. Eddy, A. Krogh, G. Mitchison (1998) : Biological sequence analysis. Cambridge University Press.

[Eis98] M. Eisen, P. Spellman, P. Brown, D. Botstein (1998) : Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95, 14863-14868.

[Enr99] A. Enright, I. Iliopoulos, N. Kyrpides, C. Ouzounis (1999) : Protein interaction maps for complete genomes based on gene fusion events. *Nature* 402, 86-90.

[Fel00] M. Fellenberg, K. Albermann, A. Zollner, H.W. Mewes, J. Hani (2000) : Integrative analysis of protein interaction data. Proceedings of the ISMB-2000 conference.

[Hen98] S. Henikoff, S. Pietrokovski & J.G. Henikoff (1998) : Superior performance in protein homology detection with the Blocks Database servers. *Nucleic Acids Research*, 26, 309-312.

[Hen99] J.G. Henikoff , S. Henikoff, S. Pietrokovski (1999) : New features of the Blocks database servers. *Nucleic Acids Research*, 27(1), 226-228.

[Hil91] Hillier, L., and Green, P. (1991) : OSP: an oligonucleotide selection program. *PCR Methods and Applications*, 1:124-128.

[Hod99] Hodges PE, McKee AH, Davis BP, Payne WE, Garrels JI (1999) : The Yeast Proteome Database (YPD) : a model for the organization and presentation of genome-wide functional data. *Nucleic Acids Research*, 27(1), 69-73.

[Hof99] K. Hofmann, P. Bucher, L. Falquet, A. Bairoch (1999) : The PROSITE database, its status in 1999. *Nucleic Acids Research*, 27(1), 215-219.

[Hol99] L. Holm, C. Sander (1999) : Protein folds and families : sequence and structure alignments. *Nucleic Acids Research*, 27(1), 244-247.

[Jon97a] Jones, S. And Thornton, J.M. (1997) : Analysis of protein-protein interaction sites using surface patches. *Journal of Molecular Biology*, Vol 272, 121-132.

[Jon97b] Jones, S. And Thornton, J.M. (1997) : Prediction of protein-protein interaction sites using patch analysis. *Journal of Molecular Biology*, Vol 272,

133-143.

[Kre99] A. Kreegipuu, N. Blom, S. Brunak (1999) : PhosphoBase, a database of phosphorylation sites : release 2.0. *Nucleic Acids Research*, 27(1), 237-239.

[Mar99a] E. Marcotte, M. Pellegrini, H. Ng, D. Rice, T. Yeates, D. Eisenberg (1999) : Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285, 751-753.

[Mar99b] Marcotte E., Pellegrini M., Thompson M., Yeates T., Eisenberg D. (1999) : A combined algorithm for genome-wide prediction of protein function. *Nature*, 402, 83-86.

[Mar01] Marcotte EM, Xenarios I, Eisenberg D. (2001) : Mining literature for protein-protein interactions. *Bioinformatics*, 17(4), 359-63.

[Mug92] S. Muggleton, R.D. King, M. J. E. Sternberg (1992) : Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, vol 5(7), 647-657.

[Pel99] Pellegrini M., Marcotte E., Thompson M., Eisenberg D., Yeates T. (1999) : Assigning protein functions by comparative genome analysis : protein phylogenetic profiles. *PNAS* 96, 4285-4288.

[San99] C. Sanchez, C. Lachaize, F. Janody, B. Bellon, L. Röder, J. Euzenat, F. Rechenmann, B. Jacq (1999) : Grasping at molecular interactions and genetic networks in *Drosophila melanogaster* using FlyNets, an internet database. *Nucleic Acids Research*, 27(1), 89-94.

[Sch95] S. Schulze-Kremer (1995) : Molecular bioinformatics : algorithms and applications. W. De Gruyter, Berlin.

[Set97] J. Setubal, J. Meidanis (1997) : Introduction to computational molecular biology. PWS publishing company.

[Spr01] E. Sprinzak, H Margalit (2001) : Correlated sequence-signatures as markers of protein-protein interaction. *Journal of Molecular Biology* 311, 681-692.

[Sri97] A. Srinivasan, R.D. King, S.H. Muggleton, M.J.E. Sternberg (1997) : Carcinogenesis predictions using ILP. In *Lecture Notes in Artificial Intelligence* 1297, Inductive Logic Programming, ILP-97.

[Ste99] L. Stein, J. Thierry-Mieg (1999) : Scriptable Access to the *Caenorhabditis elegans* Genome Sequence and other Acedb Databases. *Genome Research*, 8(12), 1308-1315.

[Ste01b] L. Stein, P. Sternberg, R. Durbin, J. Thierry-Mieg, J. Spieth (2001) : WormBase : network access to the genome and biology of *Caenorhabditis elegans*. *Nucleic Acids Research* **29(1)**:82-86.

[Swi] The SwissProt protein database, http://www.expasy.ch.

[Thi99] J. Thierry-Mieg, D. Thierry-Mieg, L. Stein (1999) : ACEDB : The ACE database manager. In S. Letovsky (ed.) : *Bioinformatics, Databases and Systems*, Kluwer Academic Publishers, 265-278.

[Thi00] Thierry-Mieg N (2000) : Protein-protein interaction prediction for *C. elegans*. Knowledge discovery in biology workshop, *4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, September 2000, Lyon.

[Thi01a] Thierry-Mieg N, Trilling L (2001) : InterDB, a prediction-oriented protein interaction database for *C. elegans*. In O. Gascuel M.F. Sagot *eds.*, *Lecture Notes in Computer Science*, 2066, 144-154, 2001.

[Wat95] M. S. Waterman (1995) : Introduction to computational biology.

Chapman & Hall.

[Whi94] James V. White, Collins M. Stultz, Temple F. Smith (1994) : Protein classification by stochastic modeling and optimal filtering of amino-acid sequences. *Mathem. Biosc.*, 119, 35-75.

[Win99] Winona C. Barker, John S. Garavelli, Peter B. McGarvey, Christopher R. Marzec, Bruce C. Orcutt, Geetha Y. Srinivasarao, Lai-Su L. Yeh, Robert S. Ledley, Hans-Werner Mewes, Friedhelm Pfeiffer, Akira Tsugita and Cathy Wu (1999) : The PIR-International Protein Sequence Database. *Nucleic Acids Research* 27(1): 39-43.

[Xen01] Xenarios I, Fernandez E, Salwinski L, Duan XJ, Thompson MJ, Marcotte EM, Eisenberg D (2001) : DIP: The Database of Interacting Proteins: 2001 update. *Nucleic Acids Research*, 29(1), 239-241.

[Yap96] T. K. Yap, O. Frieder, R. L. Martino (1996) : High performance computational methods for biological sequence analysis. Kluwer academic publishers.

[YPD] The Yeast Protein Database, http://www.proteome.com/.

Informatique

[Agr93] R. Agrawal, T. Imielinski, A. Swami (1993) : Mining association rules between sets of items in large databases. In *Proc. SIGMOD'93*, pp 207-216, ACM Press.

[Agr94] R. Agrawal, R. Srikant (1994) : Fast algorithms for mining association rules. *Proceedings if the 20th VLDB Conference*, pp 487-499.

[Agr96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. Verkamo (1996) : Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp 307-328, AAAI Press.

[Ber96] F. Bergadano, D. Gunetti (1996) : Inductive logic programming. MIT press, Cambridge, Mass.

[Bra95] I. Bratko, S. Muggleton (1995) : Applications of inductive logic programming. *Communications of the ACM*, 38(11), 65-70.

[Ben93] F. Benhamou, A. Colmerauer (1993) : Constraint logic programming : selected research. MIT press.

[Bou00] JF Boulicaut, A Bykowski (2000) : Frequent closures as a concise representation for binary data mining. In *Lecture Notes in Artificial Intelligence* 1805, PaKDD'00.

[Deh99] L. Dehaspe, H. Toivonen (1999) : Discovery of frequent Datalog patterns. *Data Mining and Knowledge Discovery* **3(1)**:7-36.

[Der95] L. De Raedt, W. Van Laer (1995) : Inductive constraint logic. In *Lecture notes in artificial intelligence*, 997, Proceedings of the fifth workshop on algorithmic learning theory.

[Fin98] P. Finn, S. Muggleton, D. Page, A. Srinivasan (1998) : Pharmacophore discovery using the inductive logic programming system Progol. *Machine learning*, 30, 241-271.

[Llo87] J. W. Lloyd (1987) : Foundations of logic programming. Springer-Verlag.

[Man94] H. Mannila, H. Toivonen, A. Verkamo (1994) : Efficient algorithms for discovering association rules. In *Proc. KDD'94*, AAAI Press.

[Man96] H. Mannila, H. Toivonen (1996) : Multiple uses of frequent sets and condensed representations. In *Proc. KDD'96*, pp 189-194, AAAI Press.

[Mug91] S. Muggleton (1991) : Inductive logic programing. *New generation computing*, 8(4), 295-318.

[Mug92] S. Muggleton, R. King, M. Sternberg (1992) : Protein secondary structure prediction using logic-based machine learning. *Protein engineering*, 5(7), 647-657.

[Mug94] S. Muggleton, L. De Raedt (1994) : Inductive logic programming : theory and methods. *Journal of logic programming*, 19(20), 629-679.

[Mug95] S. Muggleton (1995) : Inverse entailement and Progol. *New generation computing*, 13, 245-286.

[Pas99] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal (1999) : Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25-46.

[Pas00] N. Pasquier (2000) : Data mining: algorithme d'extraction et de réduction des règles d'association dans les bases de données. Thèse de doctorat, Université de Clermont-Ferrand. Janvier 2000.

[Seb97] Michèle Sebag (1997) : Distance induction in first order logic. In *Lecture*

Notes in Artificial Intelligence, 1297, Inductive logic programming ILP-97.

[Ste94] M. Sternberg, R. King, R. Lewis, S. Muggleton (1994) : Application of machine learning to structural molecular biology. *Philosophical transactions of the royal society B*, 344, 365-371.

[Tri96] Laurent Trilling (1996) : Programmation géométrique impérative et logique. Journées francophones des langages applicatifs JFLA96, 147-164.

[Tur98] M. Turcotte, S. Muggleton, M. Sternberg (1998) : Protein fold recognition. In *Lecture Notes in Artificial Intelligence*, 1446, Inductive Logic Programming ILP-98, 53-64.

Annexes

Annexe 4.1 : schéma commenté de InterDB.

```
// File models.wrm
//
// Author :    Nicolas Thierry-Mieg
// Date :      15/09/99
//
// models for my protein-protein interaction database
//
//

// Proteins are identified by the primary Swissprot Accession number

?Protein    SwissProt       SwissAC Text
                            SwissID ?SwissID XREF Protein
                            Preliminary
                            SwissOrganism ?Text
                            SwissPfamDomain ?SwissPfam XREF Proteins ?Text // status for this
domain (see explanations at end of this file)
                            SwissPrositeDomain ?SwissProsite XREF Proteins ?Text // status for
this domain
                            SwissInterpro ?SwissInterPro XREF Proteins
            PIR     PIRid ?Text
                    PIRname Text
// have to check if some proteins have several pirids or names...
// if so have to create class ?Pir (eg SwissPfam)
            GeneName ?GeneName XREF Protein
            DIPOrganism UNIQUE ?Text
            Interpro ?Interpro XREF Proteins
            Keywords ?Keyword XREF Proteins
            Localisation ?Localisation XREF Proteins
            Interactions ?Interaction
            PredictedInteractions ?PredictedInteraction
```

```
                Peptide UNIQUE ?Peptide UNIQUE Int UNIQUE Int
                YPD_similar_to ?Protein XREF YPD_similar_to Float

?SwissID        Protein ?Protein XREF SwissID

?GeneName       Protein ?Protein XREF GeneName

?SwissPfam      ID Text
                Name Text
//              Status ?Text
                Proteins ?Protein XREF SwissPfamDomain
                Predictions ?Prediction

?SwissProsite   ID Text
                Name Text
//              Status ?Text
                Proteins ?Protein XREF SwissPrositeDomain
                Predictions ?Prediction

?SwissInterpro  ID Text
                Proteins ?Protein XREF SwissInterpro

?Keyword        Name ?Text
                Proteins ?Protein XREF Keywords
                ProtsInInters Int UNIQUE ?Text
                Predictions ?Prediction

?Interpro       ID UNIQUE Text
                Name Text
                Proteins ?Protein XREF Interpro
                Predictions ?Prediction
              Type UNIQUE Domain
                     Family
                     Repeated
                     PTM
                Found_in ?Interpro
                Contains ?Interpro
                Parent ?Interpro
                Children ?Interpro
                Abstract ?Text
```

```
?Localisation          Name ?Text
                       Proteins ?Protein XREF Localisation
                       Predictions ?Prediction

?Interaction  Protein1 UNIQUE ?Protein XREF Interactions
              Protein2 UNIQUE ?Protein XREF Interactions
              Origin   DIP
                       Proteome
                       Vidal
                       Davy
              BadPIR ?Protein ?Protein
              BadGeneName ?Protein ?Protein
              Predicted ?PredictedInteraction XREF Valid

?PredictedInteraction  Protein1 UNIQUE ?Protein XREF PredictedInteractions
                       Protein2 UNIQUE ?Protein XREF PredictedInteractions
                       Valid ?Interaction XREF Predicted
                       Prediction ?Prediction

?Prediction   Protein1         PfamProt1 ?SwissPfam XREF Predictions
                               PrositeProt1 ?SwissProsite XREF Predictions
                               KeywordProt1 ?Keyword XREF Predictions
                               Interpro1 ?Interpro XREF Predictions
                               Localisation1 ?Localisation XREF Predictions
              Protein2         PfamProt2 ?SwissPfam XREF Predictions
                               PrositeProt2 ?SwissProsite XREF Predictions
                               KeywordProt2 ?Keyword XREF Predictions
                               Interpro2 ?Interpro XREF Predictions
                               Localisation2 ?Localisation XREF Predictions
              Origin UNIQUE    MatrixNic          SPECIAL
                               ObservedCount Int
                               ExpectedCount Float
                               Expected2Count Float
                               MinInter Int
                               ObsExpMinRatio Float
                               AssociationRules   ObservedFreq UNIQUE Float
                                                  ExpectedFreq UNIQUE Float
                                                  FobsFexpMinRatio UNIQUE Float
                                                  SubsetCutoffRatio UNIQUE Float
```

BadKeywordCutoff UNIQUE Int

Progol

```
// The status of a domain is (extract from swissprot manual) :
//    DR   PROSITE | PFAM; ACCESSION_NUMBER; ENTRY_NAME; STATUS.
//
// Where 'ACCESSION_NUMBER' stands for  the accession number of the PROSITE or
// Pfam pattern, profile or HMM-profile entry; 'ENTRY_NAME' is the name of the
// entry and 'STATUS' is one of the following:
//
//    n
//    FALSE_NEG
//    PARTIAL
//    UNKNOWN_n
//
// Where  'n'  is  the number  of  hits  of the  pattern  or  profile in  that
// particular protein  sequence. The  'FALSE_NEG' status indicates  that while
// the pattern or profile  did not detect the protein sequence, it is a member
// of that  particular family  or domain. The 'PARTIAL'  status indicates that
// the pattern or profile did not detect the sequence because that sequence is
// not complete and lacks the region on which is the pattern/profile is based.
// Finally the  'UNKNOWN' status  indicates uncertainties as to  the fact that
// the  sequence  is  a  member of  the  family  or  domain  described by  the
// pattern/profile. Pfam  cross-references do not make  use of the 'FALSE_NEG'
// and 'UNKNOWN' status.
//
// Examples of PROSITE and Pfam cross-references:
//
//    DR   PROSITE; PS00107; PROTEIN_KINASE_ATP; 1.
//    DR   PROSITE; PS00028; ZINC_FINGER_C2H2; 6.
//    DR   PROSITE; PS00237; G_PROTEIN_RECEPTOR; FALSE_NEG.
//    DR   PROSITE; PS01128; SHIKIMATE_KINASE; PARTIAL.
//    DR   PROSITE; PS00383; TYR_PHOSPHATASE_1; UNKNOWN_1.
//
//    DR   PFAM; PF00017; SH2; 1.
//    DR   PFAM; PF00008; EGF; 8.
//    DR   PFAM; PF00595; PDZ; PARTIAL.
//
// END OF FILE
```

Annexe 4.2 : programmes de construction de InterDB.

```
#!/usr/bin/env perl
#
# File : swiss2ace.perl
# Author : Nicolas Thierry-Mieg
# Date : 28/09/99
#

# this script takes in STDIN the sprot.dat file containing the SwissProt
# database.
# it outputs to STDOUT an ace file containing some of this info,
# formatted according to my acedb model for interacting proteins.
#
# 26/10/00 : It now also accesses file $locations_file (default ./sprotCC.location)
# to construct an array of frequent localisations (@locations), and stores
# the cellular localisation of swissprot entries when the stated localisation
# is frequent (see file sprotCC.location.perl for details).

# 13/11/00 : DEPRECATED (see 01/12/00)
# there is now a class ?Keyword, so makeKW.perl should be called
# on the output of this script (for example), to generate a kwname.ace file
# that will create the name info for all keywords (see makeKW.perl for more info).

# 01/12/00 : now call fillKwIpro.perl after parsing trembl and swissprot
# files to fill some tags, instead of makeKW.perl


$locations_file = './SwissProt.Unzipped/sprotCC.location.nocount' ;

open (LOC, $locations_file) || die "can't open file $locations_file" ;
while (<LOC>)
{
      chop ;
      my $loctmp = $_ ;
      push(@locations, $loctmp) ;
```

```
}

# global variable inCCloc for localization info flag (true iff
# program is currently inside a CC-LOCATION bloc)
$inCCloc = 0 ;
$location ="" ;

while(<>)
{
   chop ;
   if (/^\/\//)
   {
      $proteinid = shift(@acs) ;
      print "Protein $proteinid\n" ;
      print "SwissID $id\n";
      print "SwissAC $proteinid\n" ; # primary swissprot accession number

      while (@acs)  # other swissprot acs
      {
         $ac = shift(@acs) ;
         print "SwissAC $ac\n" ;
      }

      while (@genes)
      {
         $gene = shift(@genes) ;
         print "GeneName $gene\n" ;
      }

      $organism = "" ;
      while (@org)
      {
         $org = shift(@org) ;
         $organism = "$organism $org" ;
      }
      if ($organism)
      {
         $organism =~ s/\.$// ;
         $organism =~ s/^ // ;
         while ($organism =~ /, /)
```

```
        {
                ($organism =~ s/^([^,]+), and //) || ($organism =~ s/^([^,]+), //);
                $singleorganism = $1 ;
                print "SwissOrganism \"$singleorganism\"\n" ;
        }
        print "SwissOrganism \"$organism\"\n" ;
    }

#CC
    if ($location)
    {
        $inCCloc = 0 ; # just in case the last line of the current protein
                # was part of a CC - LOC block...

# remove trailing . from locations
        chop $location ;

        my $loc2 = $location ;

# $loc2 : version de $location utilisable dans un grep
        $loc2 =~ s/\(/\\\(/g ;
        $loc2 =~ s/\)/\\\)/g ;
        $loc2 =~ s/\./\\\./g ;
        $loc2 =~ s/\+/\\\+/g ;
        $loc2 =~ s/\?/\\\?/g ;
        $loc2 =~ s/\*/\\\*/g ;
        $loc2 =~ s/\[/\\\[/g ;
        $loc2 =~ s/\]/\\\]/g ;

        if (grep(/^$loc2$/, @locations))
        {
                print "Localisation \"$location\"\n" ;
        }
        $location = "" ;
    }

    while (@pir)
    {
        $pirid = shift(@pir) ;
        $pirname = shift(@pir) ;
```

```
        print "PIRid $pirid\nPIRname $pirname\n" ;
    }


    while (@kws)
    {
        $kw = shift(@kws) ;
        print "Keywords \"$kw\"\n" ;
    }

#FT

    print "\n" ;

    $sequence = "" ;
    while (@seq)
    {
        $seq = shift(@seq) ;
        $sequence = "$sequence$seq\n" ;
    }
    if ($sequence)
    {
        print "Peptide : \"$proteinid\"\n$sequence\n" ;
    }

    while (@pfam)
    {
        $pfamid = shift(@pfam) ;
        $pfamname = shift(@pfam) ;
        $pfamstatus = shift(@pfam) ;

        print "SwissPfam $pfamid\nID $pfamid\nName $pfamname\n\n" ;
        print "Protein $proteinid\nSwissPfamDomain $pfamid $pfamstatus\n\n" ;
    }

    while (@prosite)
    {
        $prositeid = shift(@prosite) ;
        $prositename = shift(@prosite) ;
        $prositestatus = shift(@prosite) ;
```

```
        print "SwissProsite $prositeid\nID $prositeid\nName $prositename\n\n" ;
        print "Protein $proteinid\nSwissPrositeDomain $prositeid $prositestatus\n\n" ;
      }

      if (@interpro)
      {
        print "Protein $proteinid\n" ;
        while (@interpro)
        {
              $interproid = shift(@interpro) ;
#       print "SwissInterpro $interproid\nID $interproid\n\n" ;
              print "SwissInterpro $interproid\n" ;
        }
        print "\n" ;
      }

      next ;
    }

    /^ID\s+(\S+)/ && ($id = $1) && next ;

    if (/^AC\s+(\w.+)/)
    {
#     @acs = () ;   inutile car a l'affichage on shift(@acs)
      $ac = $1 ;
      while ($ac)
      {
        $ac =~ s/ ?(\S+);// ;
        push(@acs, $1) ;
      }
      next ;
    }

    if (/^GN\s+(\w.+)\./)
    {
      $gene = "$1 " ;
      while ($gene)
      {
        $gene =~ s/^(\S+) // ;
        if ($1 eq "AND")
```

```
        {
                next ;
        }
        if ($1 eq "OR")
        {
                next ;
        }
        push(@genes, $1) ;
      }
      next ;
    }

    /^OS\s+(\w.+)/ && (push(@org, $1)) && next ;

# CC line
    if (/^CC   -!- SUBCELLULAR LOCATION: (.+)/)
    {
      $inCCloc = 1 ;
      $location = $1 ;
      next ;
    }
    if ($inCCloc && /^CC       (.+)/)
    {
      $location = "$location $1" ;
      next ;
    }
    elsif ($inCCloc)
    {
# we were in a CC localization block but have just exited from it
      $inCCloc = 0 ;
    }


    /^DR\s+PIR; (\S+); (\S+)\./ && (push(@pir, $1, $2)) && next ;

    /^DR\s+PFAM; (\S+); (\S+); (\S+)\./ && (push(@pfam, $1, $2, $3)) && next ;

    /^DR\s+PROSITE; (\S+); (\S+); (\S+)\./ && (push(@prosite, $1, $2, $3)) && next ;

    /^DR\s+INTERPRO; (\S+); -\./ && (push(@interpro, $1)) && next ;
```

```
   if (/^KW\s+(\w.+);$/)
   {
      $kw = "$1; " ;
      while ($kw)
      {
         $kw =~ s/^([^;]+); // ;
         push(@kws, $1) ;
      }
      next ;
   }

   if (/^KW\s+(\w.+)\.$/)
   {
      $kw = "$1; " ;
      while ($kw)
      {
         $kw =~ s/^([^;]+); // ;
         push(@kws, $1) ;
      }
      next ;
   }

# FT line : MOD_RES
# -------------------------- ----------------------------------------------
# Modification               Description
# -------------------------- ----------------------------------------------
# ACETYLATION                N-terminal or other
# AMIDATION                  Generally at the C-terminal of a mature active
#                          peptide
# BLOCKED                    Undetermined N- or C-terminal blocking group
# FORMYLATION                Of the N-terminal methionine
# GAMMA-CARBOXYGLUTAMIC ACID  Of glutamate
# HYDROXYLATION              Of asparagine, aspartic acid, proline or lysine
# METHYLATION                Generally of lysine or arginine
# PHOSPHORYLATION            Of serine, threonine, tyrosine, aspartic acid
#                          of histidine
# PYRROLIDONE CARBOXYLIC ACID N-terminal glutamate which has formed an
#                          internal cyclic lactam
# SULFATATION                Generally of tyrosine
```

```
    # -------------------------- ----------------------------------------------


        /^SQ/ && (@seq = ()) && next ;
        /^  \s+(\w.+)/ && (push(@seq, $1)) && next ;

    }
```

```
#!/usr/bin/env perl
#
# file : interpro2ace.pl
# author : NTM
#
# take in STDIN the interpro.xml file.
# outputs to STDOUT an ace file containing some info :
#  interpro id, name, and type
#  abstract without publication references
#

while (<STDIN>)
{
   chop ;

   /<interpro id="(\w+)">$/ && ($id = $1) && next ;

   /<name>(.*)<\/name>$/ && ($name = $1) && next ;

   /<type>(\w+)<\/type>$/ && ($type = $1) && next ;

   if (!($inabstract) && (/<abstract>\s+(.+)/))
   {
         $abstract = "$1" ;
         $inabstract = 1 ;
         next ;
   }
   if (/<\/abstract>/)
   {
         $inabstract = 0 ;
         next ;
```

```
}
if (($inabstract) && (/(\S.+)$/))
{
        $abstract = "$abstract $1" ;
        next ;
}

(/<found_in>/) && ($in_foundin = 1) && next ;
if (/<\/found_in>/)
{
        $in_foundin = 0 ;
        next ;
}
($in_foundin) && (/<rel_ref ipr_ref="(.+)" \/>/) && push(@foundin, $1) && next ;

(/<contains>/) && ($in_contains = 1) && next ;
if (/<\/contains>/)
{
        $in_contains = 0 ;
        next ;
}
($in_contains) && (/<rel_ref ipr_ref="(.+)" \/>/) && push(@contains, $1) && next ;

(/<parlist>/) && ($in_parent = 1) && next ;
if (/<\/parlist>/)
{
        $in_parent = 0 ;
        next ;
}
($in_parent) && (/<rel_ref ipr_ref="(.+)" type="SUBTYPE" \/>/) && push(@parent, $1) && next ;

(/<chlist>/) && ($in_child = 1) && next ;
if (/<\/chlist>/)
{
        $in_child = 0 ;
        next ;
}
($in_child) && (/<rel_ref ipr_ref="(.+)" \/>/) && push(@child, $1) && next ;
```

```
    if (/<\/interpro>/)
    {
        ($inabstract) && die "inabstract is true, it should not\n" ;
        ($in_foundin) && die "in_foundin is true, it should not\n" ;
        ($in_contains) && die "in_contains is true, it should not\n" ;
        ($in_parent) && die "in_parent is true, it should not\n" ;
        ($in_child) && die "in_child is true, it should not\n" ;

#       print STDERR "done with entry $id\n" ;

        $abstract =~ s/<[^>]+>//g ;
        $abstract =~ s/\"//g ;

        print "Interpro $id\n" ;
        print "Name \"$name\"\n" ;
        ($type eq "Repeat") && ($type = "Repeated") ;
        print "Type \"$type\"\n" ;
        my $foundin ;
        while (@foundin)
        {
           $foundin = pop (@foundin) ;
           print "Found_in $foundin\n" ;
        }
        while (@contains)
        {
           $contains = pop (@contains) ;
           print "Contains $contains\n" ;
        }
        while (@parent)
        {
           $parent = pop(@parent) ;
           print "Parent $parent\n" ;
        }
        while (@child)
        {
           $child = pop(@child) ;
           print "Children $child\n" ;
        }

        print "Abstract \"$abstract\"\n" ;
```

```
        print "\n" ;

        $id = 0 ;
        $name = "" ;
        $type = "" ;
        $abstract = "" ;

    }
}
```

```
#!/usr/bin/env perl
#
# file : ipromatch2ace.pl
# author : NTM
#
# take in STDIN the match.xml file.
# outputs to STDOUT an ace file containing the matches
#

while (<STDIN>)
{
    chop ;
    /<protein id="(\w+)"/ && ($prot = $1) && next ;
    /<interpro id="(\w+)"/ && ($ipro = $1) && next ;
    if (/<\/interpro>/)
    {
        (($prot) && ($ipro)) || die "protein is $prot, interpro is $ipro, one is empty\n" ;
        print "Protein $prot\n" ;
        print "Interpro $ipro\n" ;
        print "\n" ;
        $ipro = "" ;
        next ;
    }
    if (/<\/protein>/)
    {
        $prot = "" ;
        next ;
    }
}
```

```
#!/usr/bin/env perl

#
# File : wist2ace.pl
# Author : Nicolas Thierry-Mieg
# Date : 07/02/00
#

# grabs interaction from WISTdbs.
# it outputs to STDOUT an ace file containing this info, formatted according
# to my acedb model for interacting proteins (InterDB).

# 13/11/00 : updated for Anne Davy's interactions, where we know the
# BaitSequence rather than the BaitGene. The script can now deal with
# both forms.

# where do the ISTs come from?
# must be a valid tag for InterDB model
#$origin = "Vidal" ;
$origin = "Davy" ;


use Ace;

$InterDB = "/home/nthierry/These/InterDB3" ;

$WistDB = "/home/Acedb/Proteasome.new/Acembly.001030" ;
#$WistDB = "/home/Acedb/ISTdbPrivate" ;

$ENV{'ACEDB_NOBANNER'} = 1 ;

$istdb = Ace->connect(-program=>'tacembly', -path=>"$WistDB");

$intdb = Ace->connect(-program=>'tacembly', -path=>"$InterDB");

my $ists = $istdb->fetch_many(IST=>'*') ;

my $swissac1 ;
my $swissac2 ;
```

```
while($ist = $ists->next)
{
   my $gene1 = $ist->BaitGene ;
   my $seq1 = $ist->BaitSequence ;
   my @seqs2 = $ist->FishSequence ;

   # find $swissac1 and $swissac2, swissprot accession numbers for the genes

#    my $swissac1 = "none" ;
#    my $swissac2 = "none" ;

   if (defined $gene1)
   {
         @prot1 = $intdb->fetch(-query=>"Find GeneName $gene1 ; > Protein ; SwissOrganism =
\"Caenor*\"") ;
         # used to be :
         # ($prot1) = $intdb->fetch(-query=>"find Protein GeneName = $gene1 AND SwissOrganism =
\"Caenor*\"") ;
         # but model has changed
   }
   elsif (defined $seq1)
   {
         @prot1 = $intdb->fetch(-query=>"find GeneName $seq1 ; > Protein ; SwissOrganism =
\"Caenor*\"") ;
   }
   else
   {
         die "IST $ist has no bait!\n" ;
   }


# build the @baitacs array which holds all ACs of the bait
# (possibly more than one when several proteins correspond to a gene name)
   my @baitacs ;
   while ($prot1 = shift (@prot1))
   {
#         @swissacs1 = $prot1->get('swissAC') ;
#         $swissac1 = shift(@swissacs1) ;
         $swissac1 = $prot1->name() ;
         unshift(@baitacs, $swissac1) ;
```

```
    }

# Now look for the Fishes
    my @fishacs ;
    foreach $seq2 (@seqs2)
    {
        @prot2 = $intdb->fetch(-query=>"find GeneName $seq2 ; > Protein ; SwissOrganism =
\"Caenor*\"") ;
# NOTE : model changed, see above

        if (! @prot2)
        {
            # couldn't find a protein corresponding to $seq2
            next ;
        }

        while ($prot2 = shift (@prot2))
        {
#           @swissacs2 = $prot2->get('swissAC') ;
#           $swissac2 = shift(@swissacs2) ;
            $swissac2 = $prot2->name() ;
            push(@fishacs, $swissac2) ;
        }
    }


# finally output the interaction info for this IST
    if (@baitacs && @fishacs)
    {
        foreach $baitac (@baitacs)
        {
            foreach $fishac (@fishacs)
            {
                print "Interaction $baitac.$fishac\nProtein1 $baitac\nProtein2 $fishac\nOrigin
$origin\n\n" ;
            }
        }
    }
    else
    {
```

```
        my $bait ;
        my $baitac = "none" ;
        my $fishac = "none" ;

        if (defined $gene1)
        {
           $bait = $gene1 ;
        }
        elsif (defined $seq1)
        {
           $bait = $seq1 ;
        }
        else
        {
           die "um, this shouldn't happen, ISTs without baits were tested earlier!\n" ;
        }

        if (@baitacs)
        {
           $baitac = shift(@baitacs) ;
        }
        elsif (@fishacs)
        {
           $fishac = shift(@fishacs) ;
        }

        if (@seqs2)
        {
           foreach $seq2 (@seqs2)
           {
                print "Interaction $bait.$seq2\nBadGeneName $baitac $fishac\nOrigin $origin\n\n" ;
# NOTE : if @seqs2 contains multiple sequences and it is the bait that could
# not be found, the generated interactions will have BadGeneName none fishac
# but fishac here might not correspond to the correct seq2
# rather, it will correspond to the first element of the @seqs2 array.
# (but this shouldn't matter, those are bad interactions anyway...)
           }
        }
}
}
```

```
#!/usr/bin/env perl

#
# File : dip2ace.000308.pl
# Author : Nicolas Thierry-Mieg
# Date : 23/03/00
#

# this script takes in STDIN the dip030800.dat file containing the Database
# of Interacting Proteins (http://dip.doe-mbi.ucla.edu/), as of 8 march 2000.
# it outputs to STDOUT an ace file containing the interaction info,
# formatted according to my acedb model for interacting proteins.
# it uses the new swissprot link now available in dip.
# it has to connect to InterDB to get the swissprot accession numbers,
# since DIP uses the PIR id numbers, or swissprot IDs (but not ACs)
# to identify proteins.

use Ace;

$InterDB = "/home/nthierry/These/InterDB3" ;

$ENV{'ACEDB_NOBANNER'} = 1 ;

$intdb = Ace->connect(-program=>'tacembly', -path=>"$InterDB");

$swissid1 = "none";
$swissid2 = "none";
$organism1 = "none";
$organism2 = "none";
$pir1 = "none";
$pir2 = "none";

while(<>)
{
   chop ;
#    /^Entry ID/ && ($step=0) && next ;
   /^SWISSPROT_A\s+(\S+)$/ && ($swissid1 = $1) && next ;
   /^SWISSPROT_B\s+(\S+)$/ && ($swissid2 = $1) && next ;
   /^ORGANISM_A\s+(\w.+)$/ && ($organism1 = $1) && next ;
```

```
    /^ORGANISM_B\s+(\w.+)$/ && ($organism2 = $1) && next ;
    /^PIR_A\s+(\S+)$/ && ($pir1 = $1) && next ;
    /^PIR_B\s+(\S+)$/ && ($pir2 = $1) && next ;

    if (/^\#\#\#\#\#\#\#\#\#\#/)
    {
        $swissac1 = "none";
        if ($swissid1 ne "none")
        {
           ($prot1) = $intdb->fetch(-query=>"find SwissID $swissid1 ; >Protein") ;
        }
        elsif ($pir1 ne "none")
        {
           ($prot1) = $intdb->fetch(-query=>"find Protein PIRid = $pir1") ;
        }
        if ($prot1)
        {
           $swissac1 = $prot1->name ;
#          @swissacs1 = $prot1->get('swissAC') ;
#          $swissac1 = shift(@swissacs1) ;
        }

        $swissac2 = "none";
        if ($swissid2  ne "none")
        {
           ($prot2) = $intdb->fetch(-query=>"find SwissID $swissid2 ; >Protein") ;
        }
        elsif ($pir2 ne "none")
        {
           ($prot2) = $intdb->fetch(-query=>"find Protein PIRid = $pir2") ;
        }
        if ($prot2)
        {
           $swissac2 = $prot2->name ;
#          @swissacs2 = $prot2->get('swissAC') ;
#          $swissac2 = shift(@swissacs2) ;
        }


        if (($swissac1 ne "none") && ($swissac2 ne "none"))
```

```
        {
           print "Interaction $swissac1.$swissac2\nProtein1 $swissac1\nProtein2 $swissac2\nOrigin
DIP\n\n" ;
           print "Protein $swissac1\nDIPOrganism \"$organism1\"\n\n" ;
           print "Protein $swissac2\nDIPOrganism \"$organism2\"\n\n" ;
        }
        elsif (($pir1 ne "none") && ($pir2 ne "none"))
        {
           print "Interaction $pir1.$pir2\nBadPIR $swissac1 $swissac2\nOrigin DIP\n\n" ;

           ($swissac1 ne "none") && ( print "Protein $swissac1\nDIPOrganism \"$organism1\"\n\n" ) ;
           ($swissac2 ne "none") && ( print "Protein $swissac2\nDIPOrganism \"$organism2\"\n\n" ) ;
        }
#       else
#       {
# ici on n'a ni les 2 swissac ni les 2 pir, alors on discard l'interaction
#       }

        $swissid1 = "none";
        $swissid2 = "none";
        $organism1 = "none";
        $organism2 = "none";
        $pir1 = "none" ;
        $pir2 = "none";
        next ;
   }
}
```

Annexe 5.1 : filtre.minex.pl.

```
#!/usr/bin/perl -w

#
# file : filtre.minex.pl
#
# author : Nicolas Thierry-Mieg
#
# date : 08/01/01
#

# Adapted from filtre.pl, to take the minex output instead of freddie.

# takes in STDIN the raw output file from minex (association
# rule program from Boulicaut et al from INSA Lyon).
# (see documentation for acminer12 for details on file format).
# each line is read as a head (before the comma) and possibly
# a body (after the comma), along with the observed support.
#
# this program outputs to STDOUT a filtered version of acminer's
# output, where only some selected sets are present, along with
# the observed frequency (=~support) and the "expected" frequency
# for these sets, as follows :
# 1. sets where the head contains a "bad" descriptor (usually
# swissprot keywords) are discarded; bodies containing bad
# descriptors are cleaned by removing the bad descriptors from them.
# the (optional) list of bad descriptor names  should be in an ace file,
# its name is the second argument on the command line (currently
# named badkws.PrInt.#.ace).
# To find the correspondence between those names and the column
# numbers, the script reads the corres.number.desc file (output of
# printcorres.perl), which should be in the current dir.
#
# 2. only sets concerning both proteins are kept (i.e. containing
# attributes in both the first $totaldesc and the last $totaldesc
# columns).
```

```
#
# 3. each set S satisfying 2 is kept iff fobs > $minratio * fexp,
# where fobs is the observed frequency for S (as reported by minex),
# $minratio is a constant (currently **), and fexp is the "expected"
# frequency for set S, assuming there is no particular affinity
# between the descriptors of the 2 proteins implicit in set S.
# Specifically, fexp = f1 * f2, where f1 and f2 are the respective
# supports for the subsets of S which concern proteins 1 and 2, except
# if S is symmetrical (the descriptors concerning proteins 1 and 2 are
# the same), in which case fexp = 2 * (f1*f2), to account for the
# duplication of input lines.
# To do this, we store the f1 and f2 values in an associative array
# on a first pass using only lines where the head deals with a single
# protein, then work on lines dealing with both proteins in a second
# pass.
# This second pass uses a temp file called $tempfile, in which lines
# dealing with both proteins are stored in a preprocessed format.
# (this temp file is rm'd if $debug is off).

# 27/01/01 : output is now sorted by increasing number of descriptors
# involved in the rule before output (using @sortedresult), to allow
# filtre.2.pl to work



# $debug is for testing
#my $debug = 1 ;
my $debug = 0 ;

my $corresp = "corres.number.desc" ;

my $tempfile = "/tmp/_minextmp" ;

#total number of descriptors
my $totaldesc = 1170 ;

# minimum ratio between observed and expected frequencies to consider
# the frequent set as meaningful
#$minratio = 10 ;
my $minratio ;
# NOTE : the value of $minratio must now be passed on the command line.
```

```
# total number of lines in the matrix (read from the minex file, first data line)
my $totallines ;

# @sortedresult is used to output rules in increasing number of total descriptors involved
my @sortedresult ;

# %printed is to avoid printing out several times the same line to TEMPFILE
my %printed ;

# associative array %badkwnumbers stores the bad keywords (and possibly
# other descriptors) as keys. values are always 1
my %badkwnumbers ;

# assoc array to hold the observed frequencies of sets concerning only one protein.
# initialized in pass 1 and used in pass 2
# format is : $frequency{"@head"} = "@body:$freq", where @head is the head of a set,
# @body is its body, and $freq is the observed freq for this set.
# note that "concerning only one protein" means that the head must correspond to a
# single protein, but the body can correspond to either or both.
# If the body concerns both proteins, the "set" actually represents some real frequent sets
# that concern a single protein, and some that concern both proteins (see description of
# the condensed representation of frequent sets for more info).

my %frequency ;


# if it exists, read the list of bad keywords to remove
if (($#ARGV != 0) && ($#ARGV != 1))
{
   die "syntax : filtre.pl N [badkws.ace] < minex.result > result.filtre\nwhere N is the value of
\$minratio,\nand badkws.ace (optional) holds the list of bad keywords to be removed from frequent
sets.\n" ;
}
else
{
   $minratio = $ARGV[0] ;
   if ($#ARGV == 1)
   {
```

```
        my $badkwsfile = $ARGV[1] ;

        my %corres ; # %corres is local to this block, and will be freed soon
        open(CORRES, "$corresp") || die "cannot open file $corresp\n" ;
        # initialize the %corres associative array
        while (<CORRES>)
        {
           chop ;
           /^(\d+)\t(.*)$/ || die "cannot read line\n$_\nfrom $corresp\n" ;
           $corres{$2} = $1 ;
        }
        close CORRES ;

        open(BADKWS, "$badkwsfile") || die "cannot open file $badkwsfile\n";
        while(<BADKWS>)
        {
           chop ;
           /^Keyword : \"(.*)\"$/ || next ;
           my $kw = $1 ;
           $kw =~ s|\\\/|\/| ; # to solve pb : when acedb dumps a list of
                               # names and a name has a /, it replaces it by \/
           my $kwnb = $corres{$kw} ;
           ($debug) && print "kw $kw, kwnb $kwnb\n" ;
           $badkwnumbers{$kwnb} = 1 ;
        }
        close BADKWS ;
   }
}

sub hasbadkws
{
   # function, input : pointer to array of descriptor numbers, output 0 or 1.
   # returns 1 if the array of descriptor numbers passed as argument
   # contains a bad keyword
   my ($colsref) = @_ ;

   foreach my $col (@$colsref)
   {
        if ((defined $badkwnumbers{$col}) || (defined $badkwnumbers{($col-$totaldesc)}))
        {
```

```
            return 1 ;
        }
    }
    return 0 ;
}

sub rmbadkws
{
    # procedure, input : (pointer to) array of descriptors, the array is
    # modified by removing bad descriptors.
    my ($colsref) = @_ ;

    foreach my $i (reverse 0 .. $#$colsref)
    {
        if ((defined $badkwnumbers{$$colsref[$i]}) || (defined $badkwnumbers{($$colsref[$i]-
$totaldesc)}))
        {
            splice(@$colsref, $i, 1) ;
        }
    }
}
sub NUMER { $a <=> $b ; } # for sort below

open (TEMPFILE, ">$tempfile") || die "cannot open $tempfile for write access\n" ;

while(<STDIN>)
{
    chop ;
    chop ; # twice for DOS file
    ($debug) && print "looking at line $_\n" ;
    /^(.*):([0-9]+)$/ || next ;

    my $columns = $1 ;
    my $freq = $2 ;

    ($debug) && print "found a line!\n" ;

    if (! $columns)
        # nothing before :NNN, this is the first line giving the number
        # of lines in the matrix
```

```
    {
        $totallines = $freq ;
        ($debug) && print "total lines : $totallines\n" ;
        next ;
    }

    $columns =~ s/_// ;

# find the head and body of current line (ie before and after the comma),
# and change the scalar strings to arrays
    $columns =~ s/^([^,]+)// || die "cannot find head of line $columns\n" ;
    @head = split(/ +/, $1) ;

    $columns =~ s/, // ;
    @body = sort (NUMER split(/ +/, $columns)) ;


# deal with lines containing bad keywords (if asked to)
    if (defined(%badkwnumbers))
    {
        if (&hasbadkws(\@head))
        {
           # skip this line if head contains a bad kw
           next ;
        }
        # else remove bad keywords from body
        &rmbadkws(\@body) ;
    }



# find first and last columns in current line
    my $first = $head[0] ;
    if (@body)
    {
        ($body[0] < $first) && ($first = $body[0]) ;
    }

    my $last = $head[$#head] ;
    if (@body)
```

```
    {
        ($body[$#body] > $last) && ($last = $body[$#body]) ;
    }

    ($debug) && (print "premier $first\tdernier $last\n") ;


    if ($first > $totaldesc || $last <= $totaldesc)
        # the whole line deals with a single protein
    {
        my $line = "@body:$freq" ;
        ($debug) && (print "current line deals with a single prot : @head,$line\n") ;
        $frequency{"@head"} = $line ;
    }
    else
        # 2 proteins involved, save it in tempfile
    {
        my @toprint = sort (NUMER @head, @body) ;
        (! defined $printed{"@toprint"}) && (print TEMPFILE "@toprint:$freq\n") ;
        $printed{"@toprint"} = 1 ;

        if ($head[0] > $totaldesc || $head[$#head] <= $totaldesc)
            # the head deals with a single prot, we must both
            # save it in tempfile for pass 2 and record it in %frequency
        {
            my $line = "@body:$freq" ;
            $frequency{"@head"} = $line ;
        }
    }
# end of pass 1
}

close TEMPFILE ;

# for pass 2, finding the freq of an itemset corresponding to a single prot
sub indices_suivants
{
    # entree : (ref a) liste d'indices, et indice max qu'on peut considerer
    # effet : modifie la liste d'indices (en l'incrementant de 1 en booleen)
    my ($indicesref, $indicemax) = @_ ;
```

```
    my $compte_indices = 0 ;

    if ($#$indicesref == $indicemax)
    {
        die "all subsets of the protein have been tested, none have been found, shouldnt happen\n" ;
    }

    ($debug) && print "in indices_suivants, indices is @$indicesref\n" ;
    while (@$indicesref)
    {
        my $val = shift (@$indicesref) ;
        if ($val > $compte_indices)
        {
            unshift(@$indicesref, $compte_indices, $val) ;
            ($debug) && print "in indices_suivants indices is now @$indicesref\n" ;
            return 1 ;
        }
        else
        {
            $compte_indices++ ;
        }
    }
    ($debug) && print "indices is @$indicesref\ncompte_indices is $compte_indices\n" ;
    unshift(@$indicesref, $compte_indices) ;
    ($debug) && print "indices is now @$indicesref\n\n" ;
    return 1 ;
}

# begin pass 2 : read the $tempfile and parse it
open (TEMPFILE, "$tempfile") || die "cannot open $tempfile for read access\n" ;

while(<TEMPFILE>)
{
    chop ;
    /^([^:]*):([\d]+)/ || die "cannot parse line $_ from tempfile\n" ;
    my $freq = $2 ;
    my @columns = split(/ +/, $1) ;

    # find $freq1 and $freq2, frequencies of the subsets of current line
```

```
   # that concern protein 1 (resp 2)
   my $lastp1 = $#columns ;
   foreach my $i (1 .. $#columns) # begin with 1 because we know the first col must be from prot1
   {
         ($debug) && print "i is $i, columns(i) is $columns[$i]\n" ;
         if ($columns[$i] > $totaldesc)
         {
            $lastp1 = $i - 1 ;
            ($debug) && print "columns(i) is greater than totaldesc, lastp1 is $lastp1\n" ;
            last ;
         }
   }
   ($debug) && print "columns is @columns\nlast indice for prot1 is $lastp1, corresponding to
$columns[$lastp1]\n" ;
   my @prot1 = @columns[0 .. $lastp1] ;
   my @prot2 = @columns[$lastp1+1 .. $#columns] ;

   ($debug) && print "prot1 is therefore @prot1 and prot2 is @prot2\n" ;



# to find freq1, we look for a line (in %frequency) where head is
# a subset of prot1 and the complement (of prot1) appears in tail.

   @indices_cour = () ;

   my $freq1 ; # will hold frequency for protein1
   my $freq2 ; # idem for protein2

  combi1: while (&indices_suivants(\@indices_cour, $#prot1))
  {
     ($debug) && print "going through combi1\n" ;
     my $indicemanq = 0 ;
     my @indices_manquants = () ;

     foreach my $ind (@indices_cour)
     {
           while ($indicemanq < $ind)
           {
              push(@indices_manquants, $indicemanq) ;
              $indicemanq++ ;
```

```
        }
        $indicemanq++ ;
    }
    while ($indicemanq <= $#prot1)
    {
        push(@indices_manquants, $indicemanq) ;
        $indicemanq++ ;
    }

    my @sousprot1 = @prot1[@indices_cour] ;# @sousprot1 est le sous-ensemble de @prot1 que
l'on cherche en head
    my @manquants1 = @prot1[@indices_manquants] ;# @manquants1 idem en tail (complement de
@sousprot1)

    ($debug) && print "looking for entry with head=@sousprot1\nprot1 is @prot1\nmanquants1 is
@manquants1\n\n" ;

    if (my $line = $frequency{"@sousprot1"})
    {
        ($debug) && print "found entry with head1=@sousprot1,\tmanquants1 is @manquants1\nline
is $line\n" ;
        $line =~ /([^:]*):([\d]+)/ || die "bad format for $line in \%frequency\n" ;
        my $tail = $1 ;
        $freq1 = $2 ;
      tester_manquants1: foreach my $manque (@manquants1)
      {
        ($debug) && print "looking for $manque in $tail\n" ;
        while($tail)
        {
            $tail =~ s/(\d+) *// || die "cannot grab first descriptor from $tail\n" ;
            ($manque < $1) && next combi1 ;
            ($manque == $1) && next tester_manquants1 ;
            # sinon on cherche $manque dans la suite de $tail
        }
        # si on arrive ici on n'a pas trouvé tous les manquants
        ($debug) && print "could not find $manque\n" ;
        next combi1 ;
      }
        # ici tous les manquants ont ete trouves, c'est gagne! le resultat est dans $freq1
        ($debug) && print "found it! prot1 is @prot1, good line is $line\n\n" ;
```

```
            last combi1 ;

    }
 }

# idem pour prot2 :
   ($debug) && print "IN THE MIDDLE : freq is $freq, freq1 is $freq1\n" ;

   @indices_cour = () ;

 combi2: while (&indices_suivants(\@indices_cour, $#prot2))
 {
    my $indicemanq = 0 ;
    my @indices_manquants = () ;

    foreach my $ind (@indices_cour)
    {
           while ($indicemanq < $ind)
           {
              push(@indices_manquants, $indicemanq) ;
              $indicemanq++ ;
           }
           $indicemanq++ ;
    }
    while ($indicemanq <= $#prot2)
    {
           push(@indices_manquants, $indicemanq) ;
           $indicemanq++ ;
    }

    my @sousprot2 = @prot2[@indices_cour] ;# @sousprot2 est le sous-ensemble de @prot2 que
l'on cherche en head
    my @manquants2 = @prot2[@indices_manquants] ;# @manquants2 idem en tail (complement de
@sousprot2)

    ($debug) && print "looking for entry with head=@sousprot2\nprot2 is @prot2\nmanquants2 is
@manquants2\n\n" ;

    if (my $line = $frequency{"@sousprot2"})
    {
```

```
            ($debug) && print "found entry with head2=@sousprot2,\tmanquants2 is @manquants2\nline
is $line\n" ;
            $line =~ /([^:]*):([\d]+)/ || die "bad format for $line in \%frequency\n" ;
            my $tail = $1 ;
            $freq2 = $2 ;
          tester_manquants2: foreach my $manque (@manquants2)
          {
             while($tail)
             {
                   $tail =~ s/(\d+) *// || die "cannot grab first descriptor from $tail\n" ;
                   ($manque < $1) && next combi2 ;
                   ($manque == $1) && next tester_manquants2 ;
                   # sinon on cherche $manque dans la suite de $tail
             }
             # si on arrive ici on n'a pas trouvé tous les manquants
             next combi2 ;
          }
            # ici tous les manquants ont ete trouves, c'est gagne! le resultat est dans $freq2
            ($debug) && print "found it! prot2 is @prot2, good line is $line\n\n" ;
            last combi2 ;

     }
  }


#################################

    ($debug) && print "freq is $freq, freq1 is $freq1, freq2 is $freq2\n" ;

    my $freqexp = $freq1 * $freq2 ; # expected frequency

# dans prot2 on retire la valeur $totaldesc pour avoir reellement
# les numeros des descripteurs

          foreach (@prot2)
          {
             $_ -= $totaldesc ;
          }

# when a set is present, its symmetrical set (prot2 prot1) also is.
```

```
# therefore we discard the sets in one orientation
   if ("@prot1" lt "@prot2")
   {
         next ;
   }

   if ("@prot1" eq "@prot2")
         # multiply expected freq by 2 when the 2 proteins have
         # identical descriptors
   {
         ($debug) && print "symmetrical set @prot1\t@prot2, multiplying freqexp by 2\n" ;
         $freqexp = $freqexp * 2 ;
   }

# in truth we have been manipulating supports, not frequencies.
# this is adjusted here (divided twice for freqexp because both freq1
# and freq2 should have been divided)
   $freq = $freq / $totallines ;
   $freqexp = $freqexp / ($totallines * $totallines) ;

   if ($freq > ($freqexp * $minratio))
   {
         $sortedresult[(@prot1 + @prot2)] .= "; @prot1\t @prot2\t$freq\t$freqexp" ;
   }

#    elsif ($debug)
#    {
#         print "freq is $freq, freqexp is $freqexp, discarding this set\n(prot1 and prot2 are @prot1 and
@prot2)\n" ;
#    }
}

foreach my $listofpreds (@sortedresult)
{
   while($listofpreds)
   {
         $listofpreds =~ s/;([^;]+)// || die "cannot extract first rule from $listofpreds\n" ;
         print "$1\n" ;
   }
}
```

```
(! $debug) && (system("rm $tempfile")) ;
```

Annexe 5.2 : filtre.2.minex.iter.pl.

```
#!/usr/bin/env perl


#
# file : filtre.2.minex.iter.pl
#
# author : Nicolas Thierry-Mieg
#
# date : 31/01/01
#

# Adapted from filtre.2.minex.pl
# This is an iterative version, the recursive procedure test_eliminate
# is replaced by indices_suivants which successively generates the indices of
# all the subsets of a given itemset.


# takes in STDIN the output from filtre.minex.pl (currently result.filtre.blabla),
# and ouputs to STDOUT a more filtered version, where some sets
# have been eliminated, as follows :
# if E and F are frequent sets, and E is a subset of F, we eliminte
# E if the ratio (fobs/fexp)E is smaller than ((fobs/fexp)F/$minratio)
#

# for each pair of list of descriptors we store the scores in an
# associative array %scores of the form :
# $scores{p1d1 p1d2 p1d3:p2d1 p2d2 p2d3} = $fobs:$fexp
# where p1di are descriptors for protein 1, p2di for protein 2,
# $fobs is the observed freq for this itemset,
# and fexp the expected freq for it
# entries that must be eliminated are just undef'ed


# minimum ratio between (fobs/fexp)F and (fobs/fexp)E to keep
# both theorems (where E is a subset of F)
# can be set to infini to do no filtering
```

```
#$minratio = "infini" ;
#$minratio = 4 ;
#
# NOTE : now the value of $minratio (possibly infini) must be passed on
# the command line

if ($#ARGV != 0)
{
   die "syntax : filtre.2.pl minratio < result.filtre1 > result.filtre1.filtre2\nwhere minratio must be either an
int or the string \"infini\"\n" ;
}
else
{
   $minratio = $ARGV[0] ;
}


sub indices_suivants
{
   # entree : (ref a) liste d'indices, et indice max qu'on peut considerer
   # effet : modifie la liste d'indices (en l'incrementant de 1 en booleen)
   my ($indicesref, $indicemax) = @_ ;

   my $compte_indices = 0 ;

   if ($$indicesref[$#$indicesref] > $indicemax)
   {
         # all combinations of indices have been tested
         return 0 ;
   }

   while (@$indicesref)
   {
         my $val = shift (@$indicesref) ;
         if ($val > $compte_indices)
         {
            unshift(@$indicesref, $compte_indices, $val) ;
            return 1 ;
         }
         else
```

```
        {
            $compte_indices++ ;
        }
    }
    unshift(@$indicesref, $compte_indices) ;
}



while(<STDIN>)
{
    chop ;
    /^ ([0-9 ]+)\t ([0-9 ]+)\t(0\.[0-9]+)\t([0-9\.e-]+)$/ || die "can't read line $_" ;

    $prot1 = $1 ;
    $prot2 = $2 ;
    $fobs = $3 ;
    $fexp = $4 ;

#    print "currently considering $prot1\t$prot2\t", ($fobs /$fexp), "\n" ;

    # initialize the %scores associative array value for $prot1:$prot2
    $scores{"$prot1:$prot2"} = "$fobs:$fexp" ;



    # and then test and possibly undef all subsets of $prot1:$prot2

    # if $ratio is infini skip the following tests
    ($minratio eq "infini") && next ;

    # else get on with the tests
    @prot1array = split(' ', $prot1) ;
    @prot2array = split (' ', $prot2) ;

    my @indices_cour1 = () ;
    my @indices_cour2 = () ;

    while (&indices_suivants(\@indices_cour1, $#prot1array))
    {
        while (&indices_suivants(\@indices_cour2, $#prot2array))
```

```
        {
            my @p1tmparray = @prot1array[@indices_cour1]  ;
            my @p2tmparray = @prot2array[@indices_cour2]  ;

            my $p1tmp = join(' ', @p1tmparray) ;
            my $p2tmp = join(' ', @p2tmparray) ;

            if ($p1tmp lt $p2tmp)
            {
                if (defined $scores{"$p2tmp:$p1tmp"})
                {
                    my ($fobs2, $fexp2) = split(':', $scores{"$p2tmp:$p1tmp"}) ;

#                   print "now testing $p2tmp\t$p1tmp, which has score ",($fobs2 / $fexp2) , "\n" ;

                    if ((($fobs2 / $fexp2) * $minratio) < ($fobs / $fexp))
                    {
                        # then p2tmp:p1tmp is a bad set and should be eliminated
#                       print "now eliminating $p2tmp\t$p1tmp\n" ;
                        undef $scores{"$p2tmp:$p1tmp"} ;
                    }
                }
            }
            else
            {
                if (defined $scores{"$p1tmp:$p2tmp"})
                {
                    my ($fobs2, $fexp2) = split(':', $scores{"$p1tmp:$p2tmp"}) ;

#                   print "now testing $p1tmp\t$p2tmp, which has score ",($fobs2 / $fexp2) , "\n" ;

                    if ((($fobs2 / $fexp2) * $minratio) < ($fobs / $fexp))
                    {
                        # then p1tmp:p2tmp is a bad set and should be eliminated
#                       print "now eliminating $p1tmp\t$p2tmp\n" ;
                        undef $scores{"$p1tmp:$p2tmp"} ;
                    }
                }
            }
        }
```

```
    }
}



# now output to STDOUT the remaining live frequent sets

while (($key,$value) = each %scores)
{
    (! defined $scores{$key}) && next ;
    my ($prot1, $prot2) = split(':', $key) ;
    my ($fobs, $fexp) = split(':', $value) ;

    print "$prot1\t\t$prot2\t$fobs\t$fexp\n" ;
}
```

Annexe 5.3 : testpredictions.minex.updown.acestack.pl.

```
#!/usr/bin/perl -w

# File : testpredictions.minex.updown.acestack.pl
# Created : 01/02/2001
# Author : NTM


# Takes in STDIN a text file where each line holds the name of a
# protein, and applies a set of predictive rules (specified by
# arguments on the command line) to these proteins.
# Typically, script applied to the proteins from Anne Davy's screens,
# for validation and comparison of the various sets of predictive rules.
# outputs to STDOUT the interactions predicted.

# syntax : testpredictions.minex.pl FobsFexpMinRatio BadKwCutoff SubsetCutoffRatio <
ListOfBaits.ace > predictions

# UPDATED 01/02/01 to use descriptors as in the minex approach,
# ie swissprot keywords, interpro domains, and swissprot subcellular locations.
# Also, the algorithm to generate potentially appliable  predictive rules
# has been completely modified.
# Current algorithm : build queries with subsets (max size 3) of the descriptors from each protein.
# if the query has less than 3 descriptors, do an exact search for predictive rules
# containing exactly those descriptors and none other
# else get all predictive rules containing at least the chosen descriptors, then filter
# those to only keep truly applicable ones
# Also, we now use the ace keyset stack to optimize all queries (by
# generating a superset of possible answers to a set of queries, and saving the
# resulting keyset in the stack for further use).

#DEBUGGED 06/03/01 : regexps were not protected when grepping, this has been debugged.
# also, i now store the names of good predictive rules in an assoc array, to avoid using several
# times the same good rule.
```

```
# ADDED 13/09/01 :
# la valeur 3 qui limite le nombre de descripteurs a utiliser
# dans des requetes exactes est transformé en un paramètre optionnel
# si il n'est pas précisé la valeur choisie est 3.
$seuilStrategie = 3 ;

# use global variable $debug, if true then print extra stuff
$debug = 0 ;

$InterDB = "/home/nthierry/These/InterDB3" ;

use Ace ;



if (($#ARGV != 2) && ($#ARGV != 3))
{
   die "syntax : testpredictions.pl FobsFexpMinRatio BadKwCutoff SubsetCutoffRatio [seuil entre
strategies] < ListOfBaits.ace > predictions\n" ;
}
else
{
   $fobsfexp = $ARGV[0] ;
   $badkw = $ARGV[1] ;
   $subset = $ARGV[2] ;
   if ($#ARGV == 3)
   {
         $seuilStrategie = $ARGV[3] ;
   }
}

$intdb = Ace->connect(-program=>'tacembly', -path=>"$InterDB") || die "cannot open the database in
$InterDB\n" ;

while(<STDIN>)
{
   chop ;
   push (@baits, $_) ;
}
```

```
sub indices_suivants
{
   # entree : (ref a) liste d'indices, et indice max qu'on peut considerer
   # effet : modifie la liste d'indices (en l'incrementant de 1 en binaire)
   my ($indicesref, $indicemax) = @_ ;

   my $compte_indices = 0 ;

   if ($#$indicesref == $indicemax)
   {
         # all combinations of indices have been tested
         return 0 ;
   }

   while (@$indicesref)
   {
         my $val = shift (@$indicesref) ;
         if ($val > $compte_indices)
         {
            unshift(@$indicesref, $compte_indices, $val) ;
            return 1 ;
         }
         else
         {
            $compte_indices++ ;
         }
   }
   unshift(@$indicesref, $compte_indices) ;
}

sub mygrep
{
   # entree : une string, et la ref a une liste de strings.
   # effet : recherche la string dans la liste de maniere exacte (eq)
   # renvoie 1 si la string est trouvee, 0 sinon.
   # s'arrete des que la string est trouvee!
   my ($string, $listeref) = @_ ;

   foreach my $listeitem (@$listeref)
   {
```

```
        if ($listeitem eq $string)
        {
            return 1 ;
        }
    }
    return 0 ;
}


sub predicted_interactors
{
    # takes as argument a protein name, and returns the list of proteins
    # predicted to interact with it (using the global variables
    # $fobsfexp, $badkw and $subset to know which set of predictions
    # to use), preceded by the ?prediction name used (as a separator)

    local($baitname) = @_ ;

    my $bait = $intdb->fetch(protein=>"$baitname") ;

    my @baitkws = $bait->Keywords ;
    my @baitipros = $bait->Interpro ;
    my @baitlocs = $bait->Localisation ;
    my @baitorgs = $bait->SwissOrganism ;
    my $baitorg = shift(@baitorgs) ;
# $baitorg is the first organism for $bait, @baitorgs is the list
# of other organisms for $bait (if there are more than one)

    my %goodpreds ;
    my @goodpreds ;
# %goodpreds is an assoc array, indexed by the names of truly applicable
# predictions, to avoid redundant work
# @goodpreds will hold the corresponding prediction objects (faster than
# fetching the objects when you have their name)

    my @interactors ;
# @interactors will contain the list of predicted interactors


    ($debug) && print "bait is $bait\n" ;
```

```
# generate a superset of potentially applicable predictive rules, on which
# queries will be performed
   my $query = "Find prediction *.$fobsfexp.$subset.$badkw " ;
   foreach $kw (@baitkws)
   {
         $query .= "KeywordProt1 = \"$kw\" OR " ;
   }
   foreach $ipro (@baitipros)
   {
         $query .= "Interpro1 = $ipro OR " ;
   }
   foreach $loc (@baitlocs)
   {
         $query .= "Localisation1 = \"$loc\" OR " ;
   }

   $query =~ s/ OR $// || return ; # no descriptors for bait
# || die "bad format for initial query, should not happen...\n" ;
   $intdb->fetch(-query=>$query) ;
   $intdb->raw_query('spush') ;


   # first find applicable predictive rules

   my @indices_kws = () ;
   do
   {
         my @indices_locs = () ;
         do
         {
            my @indices_ipros = () ;
            do
            {
#                 if (($#indices_kws + $#indices_locs + $#indices_ipros) > 0)
#                 {
#                    # we will have more than 3 descriptors in this query, not good
#                    next ;
#                 }
```

```
                if (($#indices_kws + $#indices_locs + $#indices_ipros) == ($seuilStrategie - 3))
                {
                  # exactly $seuilStrategie descriptors in this query, do a "general" search,
                  # ie find all predictions containing the descriptors, then filter
                  # those to only keep truly applicable ones

# build the ace query used to find potentially applicable predictive rules :
#                 my $query = "Find prediction *.$fobsfexp.$subset.$badkw " ;
                  my $query ="" ;

                  my @kwstmp = @baitkws[@indices_kws]  ;
                  my @iprostmp = @baitipros[@indices_ipros]  ;
                  my @locstmp = @baitlocs[@indices_locs]  ;

                  foreach $kw (@kwstmp)
                  {
                        $query .= "KeywordProt1 = \"$kw\" AND " ;
                  }
                  foreach $ipro (@iprostmp)
                  {
                        $query .= "Interpro1 = $ipro AND " ;
                  }
                  foreach $loc (@locstmp)
                  {
                        $query .= "Localisation1 = \"$loc\" AND " ;
                  }

                  $query =~ s/ AND $// || die "impossible!\n" ;

                  ($debug) && print "current query is :\n$query\n" ;

                  $intdb->raw_query('spop') ;
                  $intdb->raw_query('spush') ;
                  my @currentpreds = $intdb->fetch(-query=>$query) ;

                  ($debug) && print "potentials : @currentpreds\n" ;

# now check to see if the predictive rules truly are applicable
                 test_preds: foreach $pred (@currentpreds)
                 {
```

```
            ($debug) && print "testing potential predrule $pred\n" ;
            if ($goodpreds{$pred})
            {
                ($debug) && print "$pred is good but was already found\n" ;
                next test_preds ;
            }

            foreach $descriptor ($pred->KeywordProt1)
            {
                ($debug) && print "looking for $descriptor\n" ;
#               $descriptor =~ s/(\W)/\\$1/g ; # to mask regexp metachars
#               grep(/$descriptor/, @baitkws) || next test_preds ;
                &mygrep($descriptor, \@baitkws) || next test_preds ;
                ($debug) && print "found it!\n" ;
            }
            foreach $descriptor ($pred->Interpro1)
            {
                ($debug) && print "looking for $descriptor\n" ;
#               $descriptor =~ s/(\W)/\\$1/g ; # to mask regexp metachars
#               grep(/$descriptor/, @baitipros) || next test_preds ;
                &mygrep($descriptor, \@baitipros) || next test_preds ;
                ($debug) && print "found it!\n" ;
            }
            foreach $descriptor ($pred->Localisation1)
            {
                ($debug) && print "looking for $descriptor\n" ;
#               $descriptor =~ s/(\W)/\\$1/g ; # to mask regexp metachars
#               grep(/$descriptor/, @baitlocs) || next test_preds ;
                &mygrep($descriptor, \@baitlocs) || next test_preds ;
                ($debug) && print "found it!\n" ;
            }

            # if we get here $pred is applicable
            ($debug) && print "pred $pred is good!\n" ;
            push(@goodpreds, $pred) ;
            $goodpreds{$pred} = 1 ;
          }
        }

        elsif (((($#indices_kws + $#indices_locs + $#indices_ipros) > -3) && (($#indices_kws +
```

```
$#indices_locs + $#indices_ipros) < $seuilStrategie - 3))
                    # between 1 and ($seuilStrategie-1) descriptors in the query, do an exact search,
                    # ie look for predictions containing only those descriptors
                  {
                    # build the ace query used to find applicable predictive rules :
#                   my $query = "Find prediction *.$fobsfexp.$subset.$badkw " ;
                    my $query = "" ;

                    my @kwstmp = @baitkws[@indices_kws]  ;
                    my @iprostmp = @baitipros[@indices_ipros]  ;
                    my @locstmp = @baitlocs[@indices_locs]  ;

                    foreach $kw (@kwstmp)
                    {
                          $query .= "KeywordProt1 = \"$kw\" AND " ;
                    }
                    foreach $ipro (@iprostmp)
                    {
                          $query .= "Interpro1 = $ipro AND " ;
                    }
                    foreach $loc (@locstmp)
                    {
                          $query .= "Localisation1 = \"$loc\" AND " ;
                    }

                    # restrict the predictive rule to have only the chosen descriptors and none other

                    if (@kwstmp)
                    {
                          $query .= "COUNT KeywordProt1 = " ;
                          $query .= scalar(@kwstmp) ;
                    }
                    else
                    {
                          $query .= "NOT KeywordProt1" ;
                    }
                    if (@iprostmp)
                    {
                          $query .= " AND COUNT Interpro1 = " ;
                          $query .= scalar(@iprostmp) ;
```

```
                }
                else
                {
                        $query .= " AND NOT Interpro1" ;
                }
                if (@locstmp)
                {
                        $query .= " AND COUNT Localisation1 = " ;
                        $query .= scalar(@locstmp) ;
                }
                else
                {
                        $query .= " AND NOT Localisation1" ;
                }

                ($debug) && print "current query is :\n$query\n" ;
                $intdb->raw_query('spop') ;
                $intdb->raw_query('spush') ;
                my @tmpgoodpreds = $intdb->fetch(-query=>$query) ;
                @goodpreds = (@goodpreds, @tmpgoodpreds) ;
                foreach $pred (@tmpgoodpreds)
                {
                        $goodpreds{$pred} = 1 ;
                }
              }
          }
          while (&indices_suivants(\@indices_ipros, $#baitipros)) ;
       }
       while (&indices_suivants(\@indices_locs, $#baitlocs)) ;
}
while (&indices_suivants(\@indices_kws, $#baitkws)) ;


($debug) && (print "good preds are :\n", keys(%goodpreds), "\n") ;

# now for each good prediction we have to find all predicted
# interactors of $bait, ie all proteins containing all
# descriptors (for prot2) from the prediction, and belonging
# to the same organism as $bait
```

```
   # first generate a superset of potential interactors, using the organism
   $query = "Find Protein " ;
   # must be from same organism as $bait
   $query .= "(SwissOrganism = \"$baitorg\"" ;
   foreach $org (@baitorgs)
   {
         $query .= " OR SwissOrganism = \"$org\"" ;
   }
   $query .= ")" ;
# perform query and push resulting keyset on stack
   $intdb->fetch(-query=>$query) ;
   $intdb->raw_query('spush') ;


#    foreach $predname (keys(%goodpreds))
   foreach $pred (@goodpreds)
   {
#         my $pred = $intdb->fetch(Prediction => $predname) ;
         $query = "" ;
         my @predkws = $pred->KeywordProt2 ;
         my @predipros = $pred->Interpro2 ;
         my @predlocs = $pred->Localisation2 ;

         foreach $kw (@predkws)
         {
            $query .= "Keywords = \"$kw\" AND " ;
         }
         foreach $ipro (@predipros)
         {
            $query .= "Interpro = $ipro AND " ;
         }
         foreach $loc (@predlocs)
         {
            $query .= "Localisation = \"$loc\" AND " ;
         }

         $query =~ s/ AND $// || die "bad query for finding proteins\n" ;
         # now evaluate the query and update list of answers
         $intdb->raw_query('spop') ;
         $intdb->raw_query('spush') ;
```

```
        my @newinteractors = $intdb->fetch(-query=>$query) ;
        $debug && print "query for proteins : $query\n" ;
        if (@newinteractors)
        {
            ($debug) && print "new interactors :\n@newinteractors\n" ;
            @interactors = (@interactors, "prediction$pred", @newinteractors) ;
        }
    }

    return @interactors ;
}


my @interactors ;

print "//file generated by testpredictions.minex.updown.acestack.pl @ARGV\n" ;

foreach $bait (@baits)
{
    @interactors = &predicted_interactors($bait) ;

    foreach $fish (@interactors)
    {
        if ($fish =~ /prediction(.*)/)
        {
            $pred = $1 ;
        }
        else
        {
            print "PredictedInteraction $bait.$fish\n" ;
            print "Protein1 $bait\n" ;
            print "Protein2 $fish\n" ;
            print "Prediction $pred\n\n" ;
        }
    }

}
```

Annexe 5.4 : Nombre d'interactions prédites et nombre de prédictions correctes pour chaque classe de règles.

For FexpFobs=2 , SubsetCutoffRatio=1, and BadKeywords=0 :
There are 1053 predicted interactions, of which 5 are correct.

For FexpFobs=2 , SubsetCutoffRatio=3, and BadKeywords=0 :
There are 1151 predicted interactions, of which 5 are correct.

For FexpFobs=2 , SubsetCutoffRatio=infini, and BadKeywords=0 :
There are 2371 predicted interactions, of which 5 are correct.

For FexpFobs=2 , SubsetCutoffRatio=1, and BadKeywords=50 :
There are 1644 predicted interactions, of which 15 are correct.

For FexpFobs=2 , SubsetCutoffRatio=3, and BadKeywords=50 :
There are 1865 predicted interactions, of which 17 are correct.

For FexpFobs=2 , SubsetCutoffRatio=infini, and BadKeywords=50 :
There are 3216 predicted interactions, of which 18 are correct.

For FexpFobs=2 , SubsetCutoffRatio=1, and BadKeywords=100 :
There are 2034 predicted interactions, of which 15 are correct.

For FexpFobs=2 , SubsetCutoffRatio=3, and BadKeywords=100 :
There are 2249 predicted interactions, of which 17 are correct.

For FexpFobs=2 , SubsetCutoffRatio=infini, and BadKeywords=100 :
There are 3596 predicted interactions, of which 18 are correct.

For FexpFobs=5 , SubsetCutoffRatio=1, and BadKeywords=0 :
There are 326 predicted interactions, of which 5 are correct.

For FexpFobs=5 , SubsetCutoffRatio=3, and BadKeywords=0 :
There are 326 predicted interactions, of which 5 are correct.

For FexpFobs=5 , SubsetCutoffRatio=infini, and BadKeywords=0 :
There are 731 predicted interactions, of which 5 are correct.

For FexpFobs=5 , SubsetCutoffRatio=1, and BadKeywords=50 :
There are 770 predicted interactions, of which 15 are correct.

For FexpFobs=5 , SubsetCutoffRatio=3, and BadKeywords=50 :
There are 870 predicted interactions, of which 17 are correct.

For FexpFobs=5 , SubsetCutoffRatio=infini, and BadKeywords=50 :
There are 1199 predicted interactions, of which 17 are correct.

For FexpFobs=5 , SubsetCutoffRatio=1, and BadKeywords=100 :
There are 780 predicted interactions, of which 15 are correct.

For FexpFobs=5 , SubsetCutoffRatio=3, and BadKeywords=100 :
There are 877 predicted interactions, of which 17 are correct.

For FexpFobs=5 , SubsetCutoffRatio=infini, and BadKeywords=100 :
There are 1204 predicted interactions, of which 17 are correct.

For FexpFobs=10 , SubsetCutoffRatio=1, and BadKeywords=0 :
There are 247 predicted interactions, of which 5 are correct.

For FexpFobs=10 , SubsetCutoffRatio=3, and BadKeywords=0 :
There are 247 predicted interactions, of which 5 are correct.

For FexpFobs=10 , SubsetCutoffRatio=infini, and BadKeywords=0 :
There are 247 predicted interactions, of which 5 are correct.

For FexpFobs=10 , SubsetCutoffRatio=1, and BadKeywords=50 :
There are 473 predicted interactions, of which 15 are correct.

For FexpFobs=10 , SubsetCutoffRatio=3, and BadKeywords=50 :
There are 573 predicted interactions, of which 17 are correct.

For FexpFobs=10 , SubsetCutoffRatio=infini, and BadKeywords=50 :
There are 633 predicted interactions, of which 17 are correct.

For FexpFobs=10 , SubsetCutoffRatio=1, and BadKeywords=100 :

There are 473 predicted interactions, of which 15 are correct.

For FexpFobs=10 , SubsetCutoffRatio=3, and BadKeywords=100 :
There are 573 predicted interactions, of which 17 are correct.

For FexpFobs=10 , SubsetCutoffRatio=infini, and BadKeywords=100 :
There are 633 predicted interactions, of which 17 are correct.

For FexpFobs=100 , SubsetCutoffRatio=1, and BadKeywords=0 :
There are 34 predicted interactions, of which 2 are correct.

For FexpFobs=100 , SubsetCutoffRatio=3, and BadKeywords=0 :
There are 34 predicted interactions, of which 2 are correct.

For FexpFobs=100 , SubsetCutoffRatio=infini, and BadKeywords=0 :
There are 34 predicted interactions, of which 2 are correct.

For FexpFobs=100 , SubsetCutoffRatio=1, and BadKeywords=50 :
There are 250 predicted interactions, of which 10 are correct.

For FexpFobs=100 , SubsetCutoffRatio=3, and BadKeywords=50 :
There are 250 predicted interactions, of which 10 are correct.

For FexpFobs=100 , SubsetCutoffRatio=infini, and BadKeywords=50 :
There are 250 predicted interactions, of which 10 are correct.

For FexpFobs=100 , SubsetCutoffRatio=1, and BadKeywords=100 :
There are 250 predicted interactions, of which 10 are correct.

For FexpFobs=100 , SubsetCutoffRatio=3, and BadKeywords=100 :
There are 250 predicted interactions, of which 10 are correct.

For FexpFobs=100 , SubsetCutoffRatio=infini, and BadKeywords=100 :
There are 250 predicted interactions, of which 10 are correct.