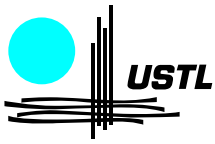


UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

U.F.R. D'I.E.E.A.

Numéro d'Ordre : 3368 Année : 2003

THÈSE

pour obtenir le grade de

DOCTEUR DE L'U.S.T.L.

DISCIPLINE : INFORMATIQUE

présentée et soutenue publiquement,

le 26 Novembre 2003, par

LAETITIA JOURDAN

Titre :

MÉTAHEURISTIQUES POUR L'EXTRACTION DE CONNAISSANCES : APPLICATION À LA GÉNOMIQUE

Jury :

Président :	Mireille Clerbout	Professeur, Université de Lille 1
Rapporteurs :	Michèle Sebag	DR CNRS, Université d'Orsay
	Gilles Venturini	Professeur, Université de Tours
Examinateurs :	Alex Freitas	Associate Professor, University of Kent at Canterbury (UK)
	Christian Dina	Ingénieur, IBL Pasteur
Directeurs :	El-Ghazali Talbi	Professeur, Université de Lille 1
	Clarisse Dhaenens	Maître de Conférence, Université de Lille 1

Table des matières

Table des figures

Remerciements

Je remercie M. El-Ghazali Talbi, directeur de cette thèse et responsable de l'équipe OPAC pour m'avoir accueillie dans son équipe et donnée les moyens de faire cette thèse dans de très bonnes conditions.

Je remercie vivement Clarisse Dhaenens, co-directeur de cette thèse, pour l'attention et l'amitié qu'elle m'a apportée durant toute ma thèse. Qu'elle soit consciente que ce travail ne serait pas là sans elle.

Mes remerciements vont ensuite aux membres du jury :

À Mme Mireille Clerbout, Professeur au LIFL qui a accepté de présider cette thèse.

À M. Gilles Venturini, Professeur à Polytech'Tours et à Mme Michèle Sebag, Directeur de Recherche CNRS au LRI d'Orsay, pour avoir accepté d'être rapporteur de cette thèse et pour les critiques constructives qui m'ont permis d'améliorer ce manuscrit.

À M. Alex Freitas, Associate Professor à l'Université de Canterbury et Christian Dina Ingénieur à l'Institut de Biologie de Lille qui m'ont fait l'honneur de participer à ce jury.

Je tiens également à remercier les membres du Laboratoire d'Informatique Fondamentale de Lille et ceux de l'équipe OPAC pour leur soutien durant ces trois années. Plus particulièrement Benjamin qui a partagé tous les moments de cette thèse en plus du bureau.

Merci également aux stagiaires pour avoir contribué à ce travail.

Merci également à mes amis qui m'ont apportée soutien et détente durant cette thèse ; Stéphane, Cécile, Benjamin, Jean-Philippe, Sylvain, Nicolas, Hélène, Caroline et tous ceux qui s'y reconnaitront.

J'aimerais remercier du fond du coeur mes parents pour leur soutien moral et matériel, et bien sûr leur relecture de mon travail. Ma famille et belle famille qui ont toujours porté un intérêt à ce que je faisais ...

Enfin, je tiens à remercier tout particulièrement Mathieu, mon mari, qui m'a soutenu et souvent réconforté durant cette thèse.

Chapitre 1

Introduction

Cette thèse s'inscrit dans le cadre des travaux de recherche en optimisation combinatoire menés par l'équipe Optimisation PArallèle et Coopérative (OPAC).

Un problème d'optimisation combinatoire est généralement caractérisé par un ensemble fini de solutions admissibles Ω et une fonction objectif $f : \Omega \rightarrow \mathbb{R}$ associant une valeur à chaque solution admissible. La résolution du problème consiste à déterminer la (ou les) solution(s) de Ω minimisant ou maximisant f. Il existe de nombreux exemples de problèmes d'optimisation combinatoire [GJ79]. Dans notre équipe, nous nous intéressons à certains d'entre eux comme la coloration des sommets d'un graphe qui consiste à affecter une couleur à chacun des sommets du graphe de manière à minimiser le nombre total de couleurs utilisées tout en assurant que deux sommets adjacents aient des couleurs différentes, le problème d'élaboration de tournées de véhicules (vehicle routing problem ou VRP) qui consiste à établir un ensemble de routes optimales qui peuvent être effectuées par une flotte de véhicules sur un ensemble de clients donnés, l'ordonnancement, le problème d'affectation quadratique (QAP), l'affectation de fréquence ... Nous nous intéressons également aux problèmes relatifs à l'extraction de connaissances et à leur modélisation en problèmes d'optimisation.

En effet, la croissance extrêmement rapide des données collectées dans les bases de données et la nécessité d'une réactivité efficace de la part des décideurs face à ces informations nouvelles ont stimulé, cette dernière décennie, le développement rapide de l'Extraction de Connaissances à partir des Données (ECD). L'ECD, ou Knowledge Discovery in Databases en anglais, est un "processus non trivial d'identification dans les bases de données de structures inconnues, valides et potentiellement exploitables" [FPSS96]. L'ECD est composée de quatre phases (voir figure 1.1) : acquisition et stockage des données (Data Warehousing), pré-traitement des données (Pre-processing), fouille de données (Data Mining) et post-traitement (Post Processing). Nos travaux se situent principalement au niveau de l'étape trois : fouille de données. Cette étape intègre à la fois le choix de la modélisation adéquate et de la méthode à utiliser ainsi que son application à la recherche de structures de données sous-jacentes et à la création de modèles explicatifs et/ou prédictifs.

Cette étape peut se décomposer en trois tâches majeures : la discrimination (classification supervisée), la catégorisation ou clustering et la recherche de règles d'association. Ces tâches

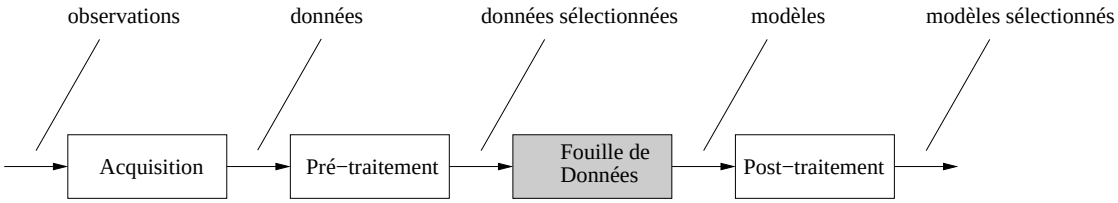


FIG. 1.1 – L'ECD à travers quatre phases

peuvent, par leur formalisation, être modélisées en problèmes d'optimisation combinatoire.

A partir de la formalisation de ces différentes tâches d'extraction de connaissances en problèmes d'optimisation combinatoire, il faut mettre en œuvre des méthodes de résolution. Habituellement, les méthodes utilisées dans ce cadre sont de trois types : les méthodes exactes, les métaheuristiques et les méthodes heuristiques spécifiques à une tâche.

Dans cette thèse, nous désirons traiter des données de grande taille issues de la génomique où de nombreux attributs sont à considérer. L'espace des connaissances potentielles étant de complexité exponentielle [ZH00], nous avons écarté les méthodes exactes. En effet, même si certaines de ces méthodes ont montré leur efficacité sur des problèmes particuliers et/ou lors de considération de critères spécifiques, nous pensons qu'elles ne permettent pas de traiter les problèmes réels nous intéressant notamment lors de la considération de critères spécifiques (liés aux applications biologiques). Ainsi, nous nous sommes plus particulièrement intéressés à la conception et à la réalisation de métaheuristiques itératives, comme les algorithmes génétiques, pour résoudre différents problèmes d'extraction de connaissances : sélection d'attributs, clustering et règles d'association. Ces trois problèmes peuvent être modélisés comme des problèmes d'optimisation NP-difficiles.
Dans le cadre de la résolution des problèmes combinatoires NP-difficiles, les algorithmes génétiques se sont montrés efficaces pour de nombreux problèmes [Mic96]. Nous nous sommes attachés à montrer leur adéquation à des problèmes d'extraction de connaissances en menant des réflexions autour de leur modélisation en problèmes d'optimisation.

Pour résoudre un tel problème en utilisant des méthodes d'optimisation, il faut, en premier lieu, définir une fonction que les solutions doivent optimiser et connaître l'espace de recherche à optimiser. Le choix de la fonction d'évaluation mesurant la qualité de la connaissance candidate à l'extraction est particulièrement important et la qualité des résultats en dépend.
Dans un premier temps, nous avons travaillé sur les fonctions d'évaluation qui peuvent être spécifiques à la tâche et/ou dépendantes au domaine d'application car, pour chaque problème d'extraction de connaissances, il existe plusieurs choix possibles. Lors de cette étape de modélisation, il a également fallu réfléchir sur la représentation d'une solution, ses avantages et inconvénients.
Cette réflexion nous a permis d'appréhender le paysage d'un problème particulier de sélection d'attributs permettant notamment de définir un schéma spécifique d'algorithme. Pour mettre en œuvre les méthodes de résolution comme les métaheuristiques, un effort particulier a été effectué dans le choix des opérateurs et des distances. De plus, nous avons étudié l'utilisation de différents opérateurs d'intensification dans une même méthode de résolution par l'application d'un choix de

mutation adaptatif. Enfin, afin de répondre à des contraintes de stabilité et pour pouvoir suivre la croissance de données, nous expliquerons les choix des techniques parallèles et hybrides introduites.

Comme support de notre étude, nous avons développé nos compétences autour de problèmes liés à la génomique en collaboration avec l'Institut de Biologie de Lille (IBL) de l'Institut Pasteur de Lille dans le cadre de la génopole de Lille. Les objectifs scientifiques de cette étude sur des problèmes réels consistent à formuler des hypothèses sur les facteurs de prédisposition à différentes pathologies multifactorielles. Ainsi, dans le cadre de notre étude avec l'IBL et plus spécialement avec le Laboratoire Génétique des Maladies Multifactorielles (LGMM), nous étudions la co-transmission d'un trait clinique ou de marqueurs génétiques afin de localiser un ou plusieurs gènes et facteurs environnementaux de prédisposition pour des maladies comme l'obésité et le diabète de type II.

Ces applications ont comme caractéristiques communes leur grand nombre d'attributs pour un relativement faible nombre d'instances. L'étude de ces problèmes nous a permis de détecter deux étapes (ou deux phases) nécessaires pour pouvoir les traiter :

- une première phase de sélection d'attributs consistant à sélectionner un ensemble de facteurs intéressants de façon à diminuer la taille de l'espace de recherche,
- une deuxième phase, soit de classification non supervisée (clustering) visant à regrouper les individus atteints par la maladie en fonction des facteurs qui semblent la provoquer, soit de recherche de règles associatives permettant d'exprimer les relations entre les facteurs et la maladie.

Ce mémoire s'articule en cinq parties.

Le **chapitre deux** présentera le cadre général de la thèse : nous introduirons les métaheuristiques à solution unique et à population de solutions, en insistant sur les algorithmes génétiques qui constituent le centre de notre travail. Nous présenterons, pour chacune de ces méthodes, leur application en fouille de données en insistant sur les différentes possibilités de codage et sur les différentes fonctions d'évaluation possibles.

Le **chapitre trois** décrira les problématiques réelles issues d'études réalisées en génomique et qui ont constitué notre cadre d'étude. Le LGMM mène des études pour analyser la co-transmission de marqueurs génétiques et des études sur le déséquilibre de liaison. Nous introduirons les notions de biologie nécessaires à la compréhension de ces études. Nous montrerons comment nous avons modélisé ces problématiques pour pouvoir les traiter. Enfin, leur étude nous a permis de réaliser un générateur de jeu de données modélisant les différentes contraintes biologiques dont nous donnerons les principales caractéristiques.

Le **chapitre quatre** présentera la première phase de notre approche. Dans un premier temps, nous présenterons la problématique de la sélection d'attributs et les méthodes classiques pour pouvoir la traiter. Pour chacune des problématiques réelles décrites dans le troisième chapitre, nous proposerons un algorithme génétique dont nous détaillerons les particularités apportées. Ces ap-

ports se situent aussi bien au niveau de la modélisation que des mécanismes qui ont été introduits pour résoudre, par optimisation, la sélection d'attributs.

Le **chapitre cinq** présentera notre approche pour traiter le partionnement de données ou clustering. Nous introduirons, tout d'abord, les notions nécessaires à la compréhension de ce problème et les différents algorithmes classiques existants dans la littérature pour le résoudre. Ces algorithmes nous serviront, lors des expérimentations, pour comparer les résultats obtenus sur différents jeux de données. Nous introduirons également une nouvelle notion de centre de cluster pour pouvoir réaliser, dans le cas nominal, une hybridation avec l'un des algorithmes les plus classiques de la littérature : Kmeans. Nous validerons notre approche sur différents jeux de données provenant de l'UCI Machine Learning Repository [BM98].

Le **chapitre six** présentera notre travail sur les règles d'associations. Nous présenterons dans un premier temps la recherche de règles d'association et les méthodes classiques de la littérature, et notamment l'algorithme Apriori, auquel nous nous comparerons dans la suite du travail. Nous présenterons ensuite les différentes fonctions d'évaluation possibles pour une règle d'association. Enfin, nous présenterons l'algorithme génétique que nous proposons en insistant sur ses opérateurs et les mécanismes adaptatifs que nous avons utilisés. Nous validerons notre approche sur une base de données classique, puis adapterons l'algorithme pour traiter une de nos problématiques biologiques.

Nous terminerons ce mémoire par différentes perspectives de recherche qui nous semblent intéressantes pour continuer ce travail.

Chapitre 2

Métaheuristiques pour l'extraction de connaissances

La majorité des problèmes d'extraction de connaissances peuvent s'exprimer comme des problèmes d'optimisation combinatoire. Or, de nombreux problèmes d'optimisation combinatoire sont NP-difficiles et ne pourront donc pas être résolus de manière exacte dans un temps "raisonnable" puisque la capacité de calcul des machines évolue linéairement alors que le temps nécessaire à la résolution de ces problèmes évolue exponentiellement. Lorsqu'on s'attaque à des problèmes réels, il faut se résoudre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé.

Au milieu des années 1970 sont apparues des méthodes qui supervisent l'évolution de solutions fournies par des heuristiques. Ces méthodes assurent un compromis entre diversification – quand il est possible de déterminer que la recherche se concentre sur de mauvaises zones de l'espace de recherche – et intensification – on recherche les meilleures solutions dans la région de l'espace de recherche en cours d'analyse. Ces algorithmes ont été appelés "métaheuristiques" et ont pour objectif de trouver des solutions dont la qualité est au-delà de ce qu'il aurait été possible de réaliser avec une simple heuristique.

Dans ce chapitre, nous introduirons différentes méthodes classiques d'optimisation (Métaheuristiques) classées en deux groupes : les méthodes à solution unique et les méthodes à population de solutions. Nous verrons, pour chaque méthode, ses différentes applications dans l'extraction de connaissances en insistant sur :

- la fonction d'évaluation,
- la représentation des solutions,
- les opérateurs.

Nous détaillerons ensuite plus précisément les algorithmes génétiques et nous ferons un bref état de l'art de leur utilisation dans les différentes tâches d'extraction de connaissances que nous étudierons dans la thèse (sélection d'attributs, catégorisation (clustering), règles d'association). En particulier, nous ne citerons pas les nombreux travaux traitant de la discrimination (classification supervisée) qui sort du cadre d'étude de cette thèse.

2.1 Les métaheuristiques à solution unique

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage [Bac99].
Nous présenterons ici les méthodes les plus utilisées et leur utilisation en extraction de connaissances : les méthodes de descente, le recuit simulé et la recherche tabou.

2.1.1 Les méthodes de descente (Hill Climbing)

Les méthodes de descente sont assez anciennes et doivent leur succès à leur rapidité et leur simplicité [Pap76, PS82]. A chaque pas de la recherche, cette méthode progresse vers une solution voisine de meilleure qualité.
La descente s'arrête quand tous les voisins candidats sont moins bons que la solution courante ; c'est-à-dire lorsqu'un optimum local est atteint.
On distingue différents types de descente en fonction de la stratégie de génération de la solution de départ et du parcours du voisinage : la descente déterministe, la descente stochastique et la descente vers le premier meilleur.

Algorithme 1 Méthode de descente générique

Procédure : φ fonction de coût
Variable locale : S solution courante
Choix d'une solution initiale S_0 ;
Solution courante $S \leftarrow S_0$;
(a.) Génération des candidats par voisinage ;
Choix du meilleur candidat C ;
if $\varphi(C) < \varphi(S)$ **then**
  $S \leftarrow C$;
  Aller en (a.) ;
end if
return S

Dans le cadre de l'extraction de connaissances, on peut citer d'une part l'algorithme des Kmeans, fortement utilisé en clustering qui est une recherche locale minimisant les distances entre les objets d'un même cluster [Fuk90], et le travail de Fränti et Kivijärvi d'autre part sur une recherche locale aléatoire pour le clustering [FK00].

2.1.2 Le recuit simulé (Simulated Annealing)

Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralisé à laquelle on introduit un paramètre de température qui sera ajusté pendant la recherche [KGV83]. Elle s'inspire des méthodes de simulation de Metropolis (années 50) en mécanique statistique.
L'analogie historique s'inspire du recuit des métaux en métallurgie : un métal refroidi trop vite

présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global.
La méthode du recuit simulé, appliquée aux problèmes d'optimisation, considère une solution initiale et recherche dans son voisinage une autre solution de façon aléatoire. L'originalité de cette méthode est qu'il est possible de se diriger vers une solution voisine de moins bonne qualité avec une probabilité non nulle. Ceci permet d'échapper aux optima locaux. Au début de l'algorithme, un paramètre T, apparenté à la température, est déterminé et décroît tout au long de l'algorithme pour tendre vers 0. De la valeur de ce paramètre va dépendre la probabilité p d'acceptation des solutions détériorantes (plus la température T est élevée, plus cette probabilité sera forte).
La performance du recuit simulé dépend, entre autres, de la règle de refroidissement (c'est à dire la décroissance du paramètre T) que l'on utilise. Un refroidissement trop rapide mènerait vers un optimum local pouvant être de très mauvaise qualité. Un refroidissement trop lent serait très coûteux en temps de calcul. Le réglage des différents paramètres (température initiale, nombre d'itérations par palier de température, décroissance de la température, ...) peut être long et difficile.

Algorithme 2 Méthode générique de recuit simulé

Procédure : φ fonction de coût, $\stackrel{P}{\leftarrow}$ affectation selon une probabilité P
Variables locales : S solution courante, T température courante, M meilleure solution, K palier de température, T_i suite de température
Choix d'une solution initiale S_0 ;
Solution courante $S \leftarrow S_0$;
Meilleure solution $M \leftarrow S$;
Température courante $T \leftarrow T_0$;
while Suivant(T,T_i) !=NULL **do**
  Itération de palier $I \leftarrow 1$
  while $I \leq K$ **do**
    Génération d'un candidat C par opération de voisinage
    $\triangle = \varphi(C) - \varphi(S)$
    if $\triangle < 0$ **then**
      $S \leftarrow C$
      if $\varphi(S) < \varphi(M)$ **then**
        $M \leftarrow S$;
      end if
    else
      Mise à jour probabilité $P \leftarrow e^{-\frac{\triangle}{kT}}$
      $S \stackrel{P}{\leftarrow} C$
    end if
    $I \leftarrow I + 1$
    $T \leftarrow$ suivant(T,T_i)
  end while
end while
return S

Dans le cadre de l'extraction de connaissances, Brown et al. [BH90] proposent SINICC (SImulation of Near-optima for Internal Clustering Criteria) pour réaliser du clustering hiérarchique. L'opérateur de perturbation utilisé est très similaire au Kmeans : il prend un point au hasard et l'affecte aléatoirement à un autre cluster. Les auteurs ont travaillé sur différentes fonctions objectives et ont appliqué SINICC à un problème réel : la surveillance radar. Dans [MBP02], les auteurs proposent d'utiliser un recuit simulé pour réaliser un clustering sur les pixels d'une image satellite de la ville de Mumbai. Dans cet algorithme, les données sont redistribuées entre les clusters de manière probabiliste.
Certains algorithmes de recuit simulé ont été utilisés dans les applications à la biologie. Ainsi, dans [LF01], Lukashin et Fuchs proposent de réaliser un clustering de données d'expression génique temporelles. Dans le cadre de la classification, Iglesia et al. [dlIWRS+03] proposent un recuit simulé pour l'extraction de connaissances en biologie. Dans [FPS00], les auteurs proposent une hybridation du recuit simulé et de la programmation génétique pour générer des arbres de décision. Dans [BK93], Boese propose d'utiliser un recuit simulé pour construire un réseau de neurones en classification supervisée.
Le recuit simulé a également été utilisé dans des hybridations avec des méthodes Tabou (voir paragraphe suivant).

L'intérêt de ces méthodes est qu'il existe une preuve de la convergence asymptotique. Ainsi, lorsque certaines conditions sont vérifiées (schéma de décroissance particulier), on a la garantie d'obtenir la solution optimale. Malheureusement, le paramétrage recommend é par la théorie n'est pas réaliste et il faut beaucoup de temps pour arriver à paramétrer ces méthodes.

2.1.3 La recherche Tabou (Tabu search)

La recherche Tabou a été introduite par F. Glover [Glo86] et a montré sa performance sur de nombreux problèmes d'optimisation. Les idées de bases de la recherche Tabou se retrouvent également dans le travail de P. Hansen [Han86]. Elle n'a aucun caractère stochastique et utilise la notion de mémoire pour éviter de tomber dans un optimum local.
Le principe de l'algorithme est le suivant : à chaque itération, le voisinage (complet ou sous-ensemble de voisinage) de la solution courante est examiné et la meilleure solution est sélectionnée. En appliquant ce principe, la méthode autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut être un meilleur voisinage.
Le risque est de cycler entre deux solutions. Pour éviter les phénomènes de cyclage, la méthode a l'interdiction de visiter une solution récemment visitée. Pour cela, une liste taboue contenant les attributs des dernières solutions visitées est tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée.
Ainsi, la recherche de la solution courante suivante se fait dans le voisinage de la solution courante actuelle sans considérer les solutions appartenant à la liste taboue.
Cette méthode ne s'arrête pas d'elle-même et il faut déterminer un critère d'arrêt en fonction du temps de recherche que l'on s'octroie. Ce critère peut être, par exemple, l'exécution d'un certain nombre d'itérations ou la non-amélioration de la meilleure solution pendant un certain nombre d'itérations. Ainsi, tout au long de l'algorithme, la meilleure solution doit être conservée car l'arrêt se fait rarement sur la meilleure solution.
Il existe de nombreuses variantes de recherches Tabou impliquant des techniques plus ou moins

sophistiquées pour intensifier ou diversifier la recherche.

Algorithme 3 Méthode générique de Tabou

Procédure : φ fonction de coût
Variables locales : S solution courante, liste tabou L, Meilleure solution M, itération courante K, nombre d'itération N
Paramétrages : Taille de la liste Tabou, Critère d'aspiration
Choix d'une solution initiale S_0 ;
Solution courante $S \leftarrow S_0$;
Meilleure solution $M \leftarrow S$;
$K \leftarrow 0$;
while $K < N$ **do**
  $K \leftarrow K + 1$
  Mise à jour de L
  Génération des candidats E par opération de voisinage
  $C \leftarrow best(E)$
  if ($\varphi(S) < \varphi(M)$) OU C n'est pas tabou OU C vérifie l'aspiration **then**
    $S \leftarrow C$
  else
    $E \leftarrow E \setminus C$
  end if
end while
return S

La recherche Tabou a été utilisée par certains auteurs pour résoudre des problèmes du clustering [CR00, AS95b].
Al-Sultan [AS95b] applique la recherche Tabou pour clusteriser des modèles. Un ensemble de solutions tests est généré à partir de la solution courante. Pour chaque exemple, un nombre aléatoire $0 \leq R \leq 1$ est généré. Si ce nombre est supérieur ou égal à une probabilité P_t, alors le modèle est changé aléatoirement de cluster (P_t est un seuil de probabilité fixé), sinon, on partitionne comme dans la meilleure solution. De la meilleure à la moins bonne solution, si le critère d'aspiration est satisfait ou une condition taboue évitée, alors la solution test est choisie comme la meilleure solution courante et chaque exemple affecté au cluster i est stocké dans la liste taboue. Si toutes les solutions tests sont taboues, alors les solutions sont régénérées à partir de la meilleure solution trouvée.
Dans [CR00], Chu et al. présentent une recherche Tabou hybridée avec un recuit simulé. Les auteurs génèrent une solution initiale pour l'approche tabou en utilisant un recuit simulé et comparent différentes méthodes d'initialisation (initialisation aléatoire, GLA (Generalized Lloyd Algorithm), Kmeans). Ils utilisent une recherche Tabou pour générer des mouvements non locaux et appliquent ensuite un recuit simulé pour sélectionner la meilleure solution pour l'itération suivante dans l'optique de réduire le temps de calcul de la génération de clusters. De plus, les solutions initiales sont générées à l'aide de GLA. Leur approche optimise la fonction de l'erreur des moindres carrés

(MSE : Mean Squared Error) :

$$MSE = \frac{1}{kT} \sum_{i=1}^{N} \sum_{j=1}^{T_i} D(X_j^{(i)}, C_i)$$

où T_i représente le nombre d'objets dans le cluster i, T le nombre total d'exemples à classer. $X_j^{(i)}$ est le $j^{\text{ème}}$ exemple du $i^{\text{ème}}$ cluster.

2.2 Les métaheuristiques à population de solutions

Les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

2.2.1 La recherche par dispersion (Scatter Search)

La recherche par dispersion est une méthode d'optimisation relativement ancienne décrite par Glover [Glo77]. Cette approche évolutionnaire a pour origine les stratégies de création de règles de décision composées et de contraintes de remplacement.
Les études récentes ont démontré les avantages pratiques de cette approche pour résoudre divers problèmes d'optimisation. La recherche par dispersion contraste avec d'autres procédures évolutionnaires, telles que les algorithmes génétiques, en utilisant des conceptions stratégiques là où d'autres approches utilisent l'aléatoire.
La recherche par dispersion opère sur un ensemble de solutions appelé l'ensemble de référence, en les combinant pour en créer des nouvelles. À la différence d'une "population" dans les algorithmes génétiques, l'ensemble de référence de solutions dans la recherche par dispersion est relativement petit.
La recherche par dispersion comprend les cinq méthodes suivantes :

- Une méthode de génération de diversification pour produire une collection de solutions diverses, en utilisant une solution d'essai arbitraire (ou solution initiale) comme entrée.
- Une méthode d'amélioration pour transformer une solution initiale en une ou plusieurs solutions d'essai améliorées.
- Une méthode de mise à jour de l'ensemble de référence pour construire et maintenir un ensemble de référence comprenant les meilleures solutions trouvées, organisé pour fournir l'accès efficace par d'autres parties de la méthode. L'incorporation de solutions à l'ensemble de référence est effectué selon leur qualité ou leur diversité.
- Une méthode de génération d'un sous-ensemble pour opérer sur l'ensemble de référence, pour produire un sous-ensemble de ces solutions comme une base pour créer des solutions combinées.
- Une méthode de combinaison de solutions pour transformer un sous-ensemble donné de solutions produit par la méthode de génération d'un sous-ensemble en un ou plusieurs vecteurs combinés de solutions.

A notre connaissance, cette méthode n'a pas été appliquée dans le domaine de l'extraction de connaissances.

2.2.2 Les algorithmes génétiques (Genetic Algorithms)

Il existe de nombreux algorithmes évolutionnaires et nous ne pouvons pas parler des algorithmes génétiques sans mentionner les méthodes de programmation évolutionnaire développée par Fogel [FOW66] et les stratégies évolutionnaires développées indépendamment par Rechenberg [Rec73] et Schwefel [Sch81]. Ils ont contribué énormément à l'intérêt porté aux algorithmes évolutionnaires.
Les algorithmes génétiques sont des méthodes basées sur les mécanismes biologiques tels que les lois de Mendel et sur le principe fondamental (sélection) de Charles Darwin [Dar59]. Holland exposa les principes de ces algorithmes pour permettre aux ordinateurs "d'imiter les êtres vivants en évoluant" pour rechercher la solution à un problème [Hol75]. Il expliqua d'abord comment ajouter de l'intelligence dans un programme informatique avec les croisements (échange du matériel génétique) et la mutation (source de la diversité génétique).

Il formalisa ensuite les principes fondamentaux des algorithmes génétiques :
- la capacité de représentations élémentaires, comme les chaînes de bits, à coder des structures complexes.
- le pouvoir de transformations élémentaires à améliorer de telles structures.

Plus récemment, Goldberg enrichit la théorie des algorithmes génétiques en s'appuyant sur le parallèle suivant [Gol89] :
- un individu est lié à un environnement par son code d'ADN,
- une solution est liée à un problème par son indice de qualité,
- une "bonne" solution à un problème donné peut être vue comme un individu susceptible de survivre dans un environnement donné.

Les algorithmes génétiques simulent le processus d'évolution d'une population. A partir d'une population de N solutions du problème représentant des individus, on applique des opérateurs simulant les interventions sur le génome telle que le croisement (cross-over) ou la mutation pour arriver à une population de solutions de mieux en mieux adaptée au problème. Cette adaptation est évaluée grâce à une fonction coût.
Nous détaillerons plus amplement ces algorithmes qui vont être la base de notre travail dans la section 2.3.

Les algorithmes génétiques ont été largement utilisés dans le cadre de l'extraction de connaissances. On les retrouve dans toutes les tâches principales. Leur application en sélection d'attributs a surtout été réalisée en optimisation mono-objectif à l'aide de méthodes enveloppantes notamment avec le classifieur K-Nearest-Neighbor [KD91, PGPD95, PDPG93, RPG+97, VdJ93], avec un réseau de neurones [OS94, YH98] ou avec des tables de décisions euclidiennes [GSW99]. En optimisation multi-objectif, les travaux sont plus rares [EHM00, YH98]. Leur application à la sélection d'attributs pour le clustering est relativement peu fréquente [KSM00].
Les algorithmes génétiques sont régulièrement utilisés pour réaliser une tâche de clustering. Les différents algorithmes proposés diffèrent aussi bien par l'approche de clustering utilisée que par leur application à des données réelles. Ainsi il existe des travaux ayant comme approche la mé-

thode des médïoides [EC00, ECM97b], la méthode des Kmeans [Col98], ou une approche hiérarchique [LL99]. Les applications sont diversifiées : données spatiales [EC00, ECM97b], données d'expression génique [Mer02] et de données de biopuces [FM01, FM02].
De nombreux auteurs ont utilisé les algorithmes génétiques dans le cadre de la recherche de règles d'association. On citera les travaux de référence de Freitas sur l'extraction de règles de classification [ALF99, FLF00, FLdA99, Fre99, Fre01, NFL99] ainsi que des travaux plus anciens [Jan93, Smi83]. Les auteurs des articles [PGI97, PGP97] expriment la recherche de règles d'association comme une extraction de patterns. Les algorithmes génétiques pour rechercher des règles d'association ont été appliqués à des problématiques réelles comme les bases de données commerciales [dlIDRS96] et la détection de fraude par analyse de séquence d'événements [Wei99].
Certains algorithmes seront détaillées dans la section 2.4.

2.2.3 Programmation génétique (Genetic Programming)

Récemment, un nouveau paradigme génétique fait son apparition, c'est la Programmation Génétique. L'idée sous-jacente à cette approche est qu'il n'est pas nécessaire d'utiliser des codages linéaires (les simples vecteurs de l'algorithme génétique traditionnel) pour soumettre des programmes à une évolution génétique. Le principal promoteur de ce paradigme est J. Koza [Koz92]. L'algorithme consiste à faire évoluer une population constituée d'un grand nombre de programmes. La plupart des algorithmes de programmation génétique travaille avec une population modélisée sous forme d'arbres. Lors de la recherche, la profondeur de ces arbres peut augmenter fortement en taille. Au départ, la population est constituée de programmes créés aléatoirement. Chaque programme est évalué selon une méthode propre au problème posé. A chaque itération (génération), on classe les programmes en fonction des notes qu'ils ont obtenues, et on créée une nouvelle population, où les meilleurs programmes auront une plus grande chance de survivre ou d'avoir des enfants que les autres. Ce principe est le même qu'en algorithmique génétique classique, mais les opérateurs de croisement et de mutation sont différents. En effet, ils travaillent directement sur la structure d'arbre du programme.

Il existe de nombreuses applications de la programmation génétique au domaine de l'extraction de connaissances. Ainsi, dans le domaine de la classification, la programmation génétique a été appliquée à la quantification de maladies [WF01], à l'identification automatique de minerais [RFY01], à la classification automatique de nouveaux textes [Mas94], à la classification d'images en l'hybridant avec des réseaux de neurones [Tac93] ainsi qu'à la bioinformatique [Han93]. Dans le cadre d'application à la classification, on retrouve notamment des travaux sur les arbres de décision [Sie94] ou de grammaire stochastique [RS03].

La programmation génétique est aussi fortement utilisée pour découvrir des règles (voir Figure 2.1). Ainsi plusieurs auteurs proposent des algorithmes pour découvrir des règles de classification ou d'association. Andre propose plusieurs algorithmes pour extraire des règles grâce à la programmation génétique dans la reconnaissance de caractères [And97] et dans des études sur l'écriture manuscrite [And94]. Dans [Bor01], l'auteur utilise la programmation génétique pour extraire des règles floues. Dans [BLF99], les auteurs proposent d'extraire des règles de classification dans le domaine médical. Dans [RP99], les auteurs proposent un algorithme pour trouver de nouvelles règles de classification afin d'améliorer la convergence d'un algorithme de réseau de

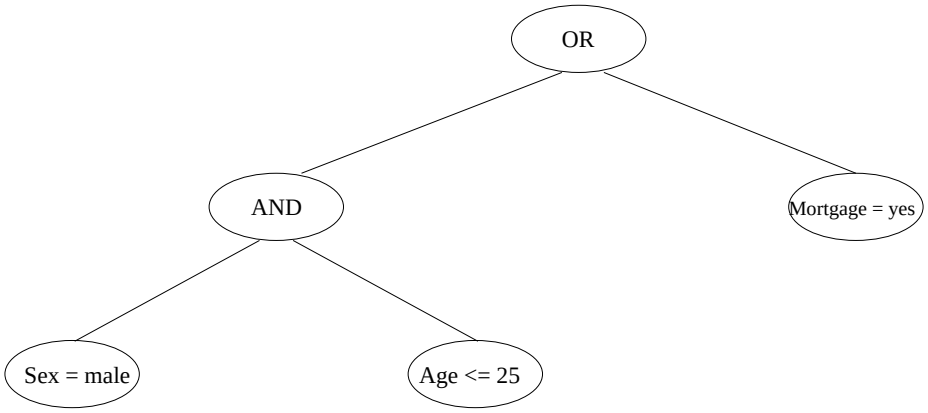


FIG. 2.1 – Programmation génétique et recherche de règles : Exemple de représentation d'un individu ayant uniquement des terminaux de type booléen. Codage de la règle IF((Sex = male) AND (Age $\leq$ 25)) OR (Mortgage = yes).

neurones de rétropropagation standard.

2.2.4 Les colonies de fourmis (Ants System)

L'histoire de l'intelligence en essaim remonte à l'étude du comportement de fourmis à la recherche de nourriture au départ de leur nid, par Goss, Deneubourg et leur équipe [DPV83, DG89]. En se déplaçant du nid à la source de nourriture et vice-versa (ce qui, dans un premier temps, se fait essentiellement d'une façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de créer une piste chimique. Les fourmis peuvent sentir ces phéromones qui ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones (voir figure 2.2). Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par d'autres fourmis pour retrouver les sources de nourriture détectées par leurs consoeurs.

Il a été démontré expérimentalement que ce comportement permet l'émergence des chemins les plus courts entre le nid et la nourriture, à condition que les pistes de phéromones soient utilisées par une colonie entière de fourmis.
Le système de fourmis (Ants System - AS) est une méthode d'optimisation basée sur ces observations proposées par Dorigo [DMC91, DMC96, Dor92]. Le système de fourmis a été employé avec succès sur des nombreux problèmes (voyageur de commerce, affectation quadratique, ...) mais les auteurs ont remarqué que l'AS n'a pas un comportement très exploratoire ce qui a conduit les auteurs à utiliser des hybridations du système de fourmis avec des recherches locales.

Les colonies de fourmis ont été utilisées en extraction de connaissances. On retrouve notamment leur utilisation pour effectuer des tâches de clustering. Ainsi, dans [HM02], Handl et Meyer proposent d'utiliser des colonies de fourmis pour regrouper des textes issus de moteur de recherche.

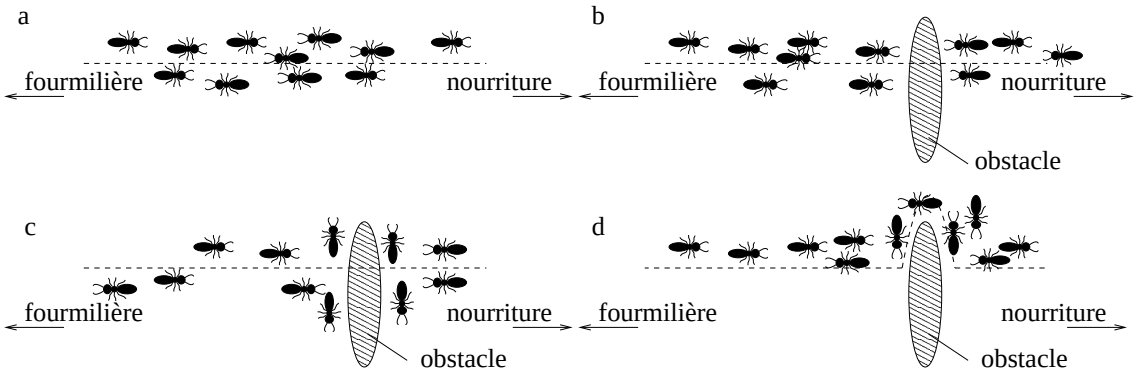


FIG. 2.2 – (a) Les fourmis suivent un chemin entre la fourmilière et la nourriture. (b) Un obstacle apparaît sur le chemin ; les fourmis choisissent entre prendre à droite et à gauche avec équiprobabilité. (c) La phéromone s'évapore sur le chemin le plus long. (d) Toutes les fourmis choisissent le chemin le plus court.

L'algorithme AntClass développé par Monmarché et al. est un algorithme de clustering hybride où la recherche du nombre de classes est effectuée par des fourmis artificielles [Mon99]. Ensuite, un algorithme classique en classification, les centres mobiles, est utilisé pour effacer les erreurs de classification inhérentes à la méthode stochastique. Les colonies de fourmis ont été aussi utilisées pour rechercher des règles, ainsi Parpinelli et al. proposent AntMiner pour rechercher des règles de classification et l'appliquent à des bases de données médicales [PLF02].

2.2.5 Algorithmes à essaim de particules (Particle Swarm Optimiser)

Les algorithmes d'optimisation par essaim de particules (PSO) ont été introduit en 1995 par Kennedy et Eberhart comme une alternative aux algorithmes génétiques standards [KE95]. Ces algorithmes sont inspirés des essaims d'insectes (ou des bancs de poissons ou des nuées d'oiseaux) et de leurs mouvements coordonnés. En effet, tout comme ces animaux se déplacent en groupe pour trouver de la nourriture ou éviter les prédateurs, les algorithmes à essaim de particules recherchent des solutions pour un problème d'optimisation. Les individus de l'algorithme sont appelés particules et la population est appelée essaim.
Dans cet algorithme, une particule décide de son prochain mouvement en fonction de sa propre expérience, qui est dans ce cas la mémoire de la meilleure position qu'elle a rencontrée, et en fonction de son meilleur voisin. Ce voisinage peut être défini spatialement en prenant par exemple la distance euclidienne entre les positions de deux particules ou sociométriquement (position dans l'essaim de l'individu). Les nouvelles vitesse et direction de la particule seront définies en fonction de trois tendances : la propension à suivre son propre chemin, sa tendance à revenir vers sa meilleure position atteinte et sa tendance à aller vers son meilleur voisin. Les algorithmes à essaim de particules peuvent s'appliquer aussi bien à des données discrètes qu'à des données continues.

Les algorithmes à essaim de particules ont été utilisés pour réaliser différentes tâches d'extraction de connaissances. Dans le cadre de la sélection d'attributs, Agrafiotis propose un algorithme basé sur les essaims de particules pour l'étude de la relation quantitative entre la structure et l'acti-

vité de composant chimique (Quantitative Structure Activity Relationship) [AC02]. Omran utilise les essaims de particules pour effectuer une classification d'images [OSE02]. Dans le cadre de l'extraction de règles de classification, les PSO ont été comparés aux algorithmes génétiques et à l'algorithme C4.5 [SNS03] et ont été appliqués à la génération de règles pour définir le profil des utilisateurs d'un site web [UB03] et à l'extraction de règles à partir d'un réseau de neurones [HWY+98]. Dans [XDE+03], les auteurs utilisent une méthode hybride basée sur les essaims de particules et l'algorithme de clustering "Self-Organizing Maps" pour réaliser un partitionnement des gènes dans des expérimentations d'expression génique.

2.2.6 Systèmes immunitaires artificiels (Artificial Immune Systems)

Les systèmes immunitaires artificiels (AIS) sont apparus dans les années 90 et sont inspirés du fonctionnement du système immunitaire humain qui est un mécanisme de défense capable d'apprendre. Beaucoup de propriétés des systèmes immunitaires ont un grand intérêt pour les informaticiens par exemple :

- chaque individu possède son propre système immunitaire avec ses forces et vulnérabilités,
- les molécules qui n'appartiennent pas à l'individu sont reconnues et éliminées,
- le système immunitaire peut détecter et réagir aux pathogènes (appelés antigène *Ag*) que le corps n'a jamais rencontrés,
- etc ...

Un des types de réponse immunitaire est la sécretion d'anticorps. Les anticorps sont des molécules réceptrices permettant de reconnaître l'antigène et de le bloquer.
Cette métaphore est utilisée par les AIS où un anticorps va représenter une solution potentielle au problème. On peut se référer aux travaux de Forrest et Kephart pour plus de détails [FPAC94, SHF00, Kep94].

Les systèmes immunitaires artificiels sont bien adaptés à l'extraction de connaissance et donnent des résultats intéressants [TK01]. Dans le cadre du clustering, on peut citer les travaux de De Castro [dCZ00] ou de Nasroui qui réalisent un clustering des flux de données sur le web et une extraction de profils d'utilisateurs [NGCD02, NGD02]. Les AIS ont également été utilisés dans le cadre de l'extraction de règles [CF01] et de nombreux auteurs les ont utilisés pour extraire des règles d'association dans le cadre de la détection de fraudes [LNY+00, Ove02]. Dans le cadre de la classification, les systèmes immunitaires artificiels ont également été appliqués [Car00, TC02].

2.2.7 Algorithmes à Estimation de Distribution (Estimation of Distribution Algorithms)

Récemment, une nouvelle classe d'algorithmes a fait son apparition : les algorithmes à estimation de distribution (EDA). Ces algorithmes ont été pour la première fois introduits en 1994 par Baluja dans ses travaux sur Population Based Incremental Learning (PBIL) [Bal94, BC95] puis en 1996 par Mühlenbein et Paaß [MP96]. Les stratégies évolutionnaires mettent en œuvre des opérateurs de mutation et de croisement mais il est difficile pour un utilisateur inexpérimenté de choisir l'opérateur approprié à son problème. Les algorithmes à estimation de distribution reprennent les principes des algorithmes à population mais utilisent des modèles probabilistes à la

place d'opérateurs de mutation et de croisement pour construire de nouveaux individus. Le modèle de fonctionnement de l'algorithme est le suivant [Yu02] :

1. Génération de la population initiale.
2. Sélection des individus prometteurs.
3. Estimation de la distribution de ces individus (construction d'un modèle probabiliste).
4. Génération de nouvelles solutions à partir du modèle probabiliste.
5. Retour en 2 jusqu'à ce que le critère d'arrêt soit satisfait.

Les EDA peuvent être appliqués aussi bien sur un domaine discret que sur un domaine continu [LL01, PGL99]. Il existe de nombreux algorithmes à estimation de distribution qui peuvent être classés selon le modèle utilisé pour la construction des nouveaux individus : compact GA (produit de distribution de Bernouilli), Population Based Incremental Learning (règle de Hebian), Univariate Marginal Distribution Algorithm, extended compact GA (produit de distribution marginale), Bayesian Optimization Algorithm (réseau bayesien), ... [Yu02].

Dans le cadre de l'extraction de connaissances, les EDA sont de plus en plus utilisés. Certains auteurs les ont utilisés pour réaliser des tâches de préparation de données comme la sélection d'attributs [ILS01a, ILS01b, CP02] et notamment son application en bioinformatique pour la prédiction des sites d'épissage [SDB+03], la sélection de gènes pour la classification de cancer [BLIS01], la sélection d'attributs pour l'étude de la survie de patients malades de la cirrhose [SLI+01]. Les EDA ont été aussi appliqués à la tâche de clustering et leur efficacité a été comparée avec l'algorithme Kmeans [RLS01]. Dans le cadre de l'induction de règles, on peut citer les travaux de Sierra [SJI+01].

2.3 Les algorithmes génétiques : présentation générale

Dans le cadre de ce travail, nous nous attacherons plus particulièrement aux algorithmes génétiques car, dans le cadre de notre équipe nous avons développé des compétences dans ce domaine. Dans cette section, nous donnerons le vocabulaire nécessaire à la compréhension des algorithmes génétiques, et nous identifierons les points importants de ces algorithmes.
Au siècle dernier, Charles Darwin observa les phénomènes naturels et fit les constatations suivantes [Dar59] :

- L'évolution n'agit pas directement sur les êtres vivants ; elle opère en réalité sur les chromosomes contenus dans leur ADN.
- L'évolution a deux composantes : la sélection et la reproduction. La sélection garantit une reproduction plus fréquente des chromosomes des êtres vivants les plus robustes.
- La reproduction est la phase durant laquelle s'effectue l'évolution.

La terminologie employée est empruntée à la génétique :

- Les chromosomes sont les éléments à partir desquels sont élaborés les solutions (individus).
- La population est l'ensemble des chromosomes.
- La reproduction est l'étape de combinaison des chromosomes. La mutation et le croisement génétiques sont des méthodes de reproduction.

D'autres notions sont propres au domaine des algorithmes génétiques :

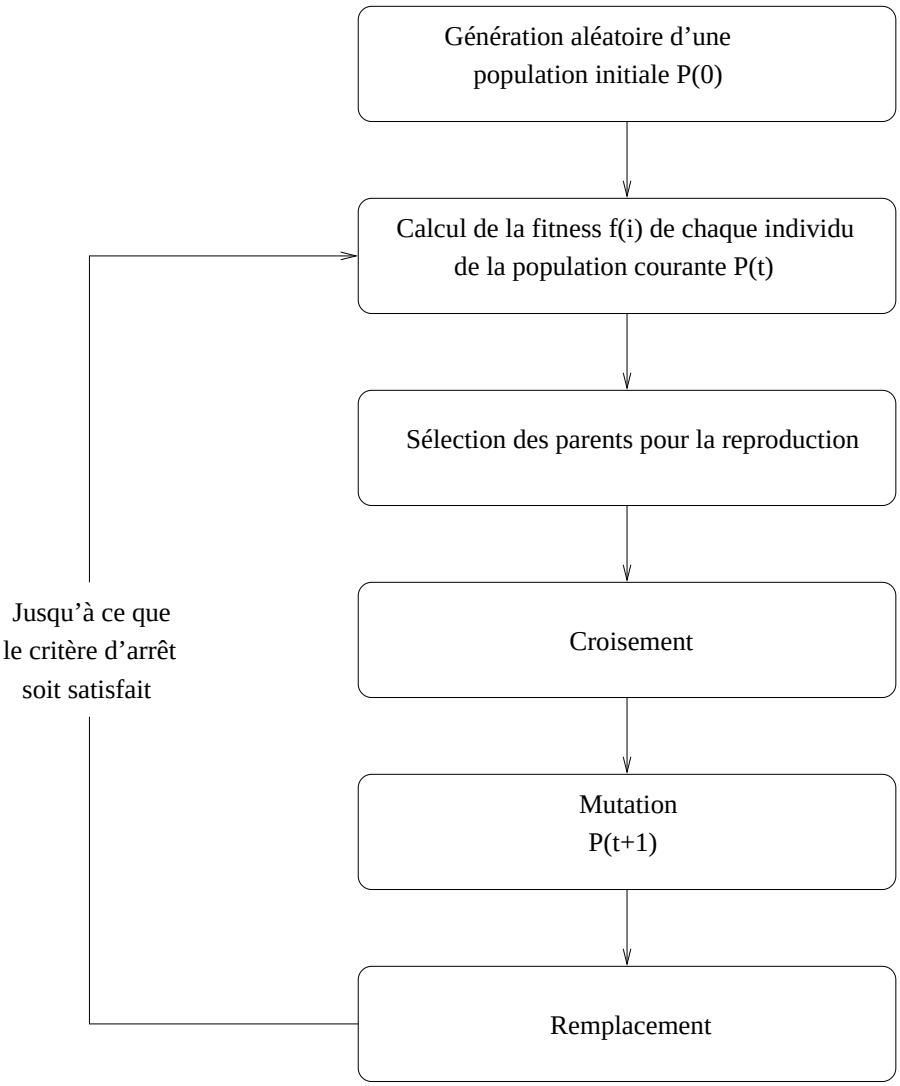


FIG. 2.3 – Fonctionnement général d'un AG de base

– L'indice de qualité (fitness), aussi appelé indice de performance, est une mesure abstraite permettant de classer les chromosomes.
– La fonction d'évaluation ou fonction coût est la formule théorique qui permet de calculer l'indice de qualité d'un chromosome.

La figure 2.3 nous montre le fonctionnement itératif simplifié d'un algorithme génétique.

Les algorithmes génétiques ont été utilisés pour résoudre un grand nombre de problèmes d'optimisation. Chaque individu, modélisé par son patrimoine génétique, peut être considéré sous deux points de vue :

– le point de vue génotypique qui décrit l'individu d'après sa représentation,
– le point de vue phénotypique qui correspond à l'expression du code génétique dans l'environnement biologique.

La figure 2.4 montre le déroulement de l'algorithme en alternance sur les plans génotypique et phénotypique.

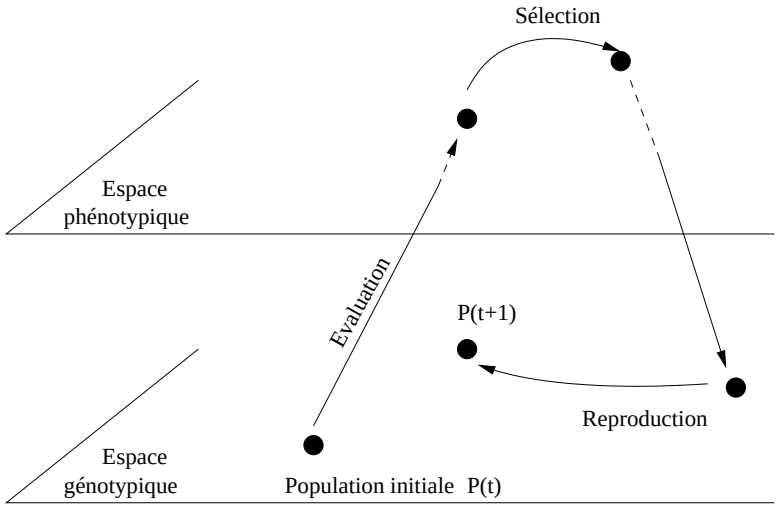


FIG. 2.4 – Déroulement d'un algorithme évolutionnaire sur les plans génotypique et phénotypique.

Pour pouvoir développer un algorithme génétique et résoudre efficacement un problème, il faut déterminer :

- Comment représenter les solutions (codage du chromosome) ?
- Quelle va être la fonction d'évaluation ?
- Quels sont les opérateurs à employer ?
- Quels sont les paramètres (taille de population, probabilité d'application des opérateurs) qui sont adaptés ?

2.3.1 Représentation

La représentation doit être *complète*, c'est à dire que toutes les solutions possibles du problème doivent pouvoir être codées à l'aide de cette représentation. En effet, si une solution ne peut pas être représentée, l'algorithme génétique ne pourra jamais la trouver.
De plus, toutes les solutions codables doivent correspondre à des solutions réalisables c'est à dire à des points de l'espace de recherche (principe de *validité*), mais il est possible d'adapter l'algorithme génétique pour éviter des solutions invalides.
La représentation peut aussi produire plusieurs chromosomes pour coder la même solution. Cette *redondance* peut, si elle est élevée, poser des problèmes pour la convergence [JB91a].
Goldberg [Gol89] donne deux principes de base pour choisir la représentation d'une solution d'un algorithme génétique. Tout d'abord, le codage doit contenir des blocs de construction contenant une information ayant du sens. Ensuite, l'alphabet doit être le plus petit permettant une expression naturelle du problème.

2.3.2 Fonction d'évaluation

La fonction d'évaluation quantifie la qualité de chaque chromosome par rapport au problème. Elle est utilisée pour sélectionner les chromosomes pour la reproduction. Les chromosomes ayant une bonne qualité ont plus de chance d'être sélectionnés pour la reproduction et donc plus de chance que la population suivante hérite de leur matériel génétique. La fonction d'évaluation produit la pression qui permet de faire évoluer la population de l'algorithme génétique vers des individus de meilleur qualité. Clairement, le choix de la fonction d'évaluation va fortement influer sur le succès de l'algorithme génétique.

2.3.3 Opérateurs génétiques

Les opérateurs génétiques travaillent directement sur les individus composant la population. On distingue différents opérateurs : opérateur d'initialisation, opérateur de sélection, opérateur de croisement et opérateur de mutation. Ils permettent, notamment dans le cas de la mutation ou du croisement, de créer des nouvelles solutions.

2.3.3.1 L'opérateur d'initialisation

Cet opérateur est utilisé pour générer la population initiale de l'algorithme génétique. La population initiale doit contenir des chromosomes qui soient bien répartis dans l'espace des solutions pour fournir à l'algorithme génétique un matériel génétique varié. La façon la plus simple est de générer aléatoirement les chromosomes.

2.3.3.2 L'opérateur de sélection

La sélection tend à augmenter l'importance des bonnes solutions par rapport aux mauvaises. C'est une heuristique utilisée par l'algorithme génétique : les bonnes solutions sont supposées être les plus prometteuses pour la génération de descendants [Ven96] (cf Fig 2.5).
Il existe plusieurs méthodes de sélection. Les plus connues sont la sélection proportionnelle à la fonction fitness, la sélection sur le rang et la sélection en tournoi.

Sélection sur la fitness
La sélection sur la fitness (ou sélection par proportionnalité) est en général implémentée comme une simulation d'une roulette biaisée [Gol89]. Chaque individu est représenté par une partie de la roue de taille proportionnelle à sa fitness. Ainsi un individu c_i a la probabilité suivante d'être sélectionné :

$$P_{Sélection}(c_i) = \frac{F_{évaluation}(c_i)}{\sum_{j=1}^{n} F_{évaluation}(c_j)}$$

où n représente la taille de la population de l'algorithme génétique.
Comme on veut sélectionner $nbselect = n \times taux\ de\ sélection$ chromosomes en moyenne :

$$P_{Sélection}(c_i) = \frac{F_{évaluation}(c_i)}{\sum_{j=1}^{n} F_{évaluation}(c_j)} \times nbselect$$

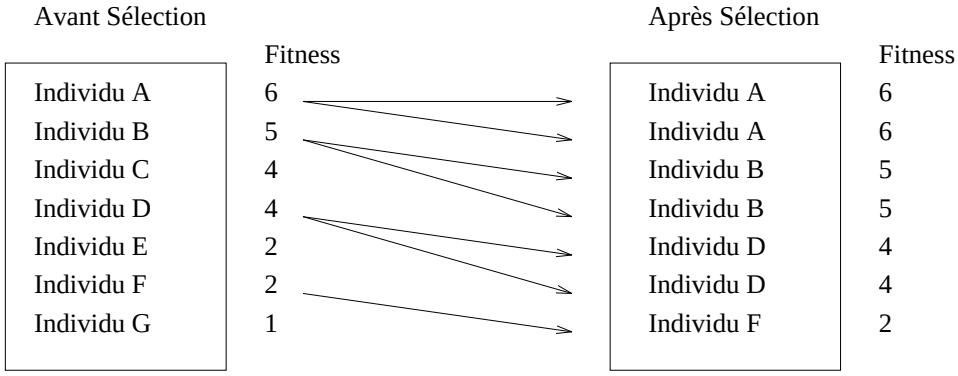


FIG. 2.5 – Exemple de sélection sur une population fictive.

Cette méthode a l'avantage d'être simple et intuitive mais elle présente certains désavantages [Deb00]. Elle requiert, par exemple, un temps de calcul important pour calculer la somme des qualités.

Sélection sur le rang
Cette méthode de sélection est divisée en deux étapes [Whi89, Gre00]. Tout d'abord, il faut ranger les individus par ordre croissant (ou décroissant pour un problème de maximisation) de leur qualité. Ensuite, une procédure de sélection similaire à la sélection sur la fitness mais appliquée au rang est utilisée. Cette procédure permet d'attribuer une probabilité de sélection p_i selon leur rang ("Ranking selection") :

$N.p_i = F - (ri - 1) \times \frac{2F-2}{N-1}$ est le nombre moyen d'enfants de l'individu i.

Où ri est le rang de i, N est le nombre d'individus et F est la "pression de sélection". Ce type de sélection représente un changement d'échelle dynamique non-linéaire pour le critère de qualité. L'un des désavantages de ce type de sélection vient du fait qu'il faille classer les individus ce qui peut gêner certaines stratégies de parallélisation.

Sélection par Tournoi
On sélectionne $\frac{n}{2}$ individus aptes à la reproduction. Ces individus sont sélectionnés de la manière suivante :
K individus sont tirés au sort dans la population des n individus (K est un paramètre appelé taille du tournoi). Il existe différentes sélections par tournoi : déterministe ou probabiliste. Dans le cas du tournoi déterministe, le meilleur des K individus gagne le tournoi. Dans le cas probabiliste, chaque individu peut être choisi comme vainqueur avec une probabilité proportionnelle à sa fonction d'évaluation.
Il est tout à fait possible que certains individus participent à plusieurs tournois : s'ils gagnent plusieurs fois, ils auront donc droit d'être copiés plusieurs fois dans la génération intermédiaire, ce qui favorisera la pérennité de leurs gènes. La figure 2.6 nous montre un exemple de sélection par tournoi entre deux individus (tournoi binaire).

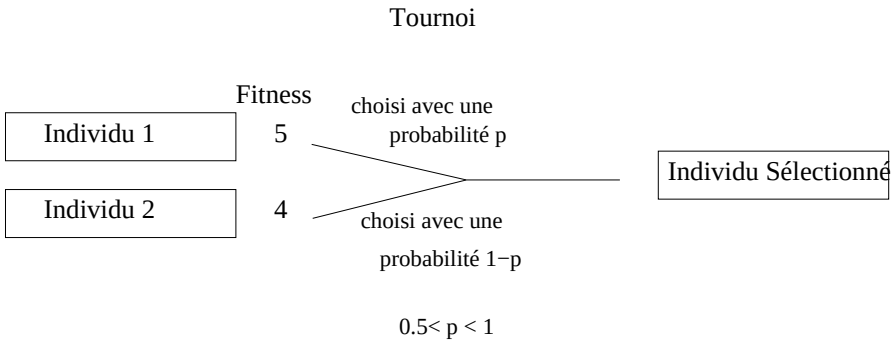


FIG. 2.6 – Exemple de sélection par Tournoi.

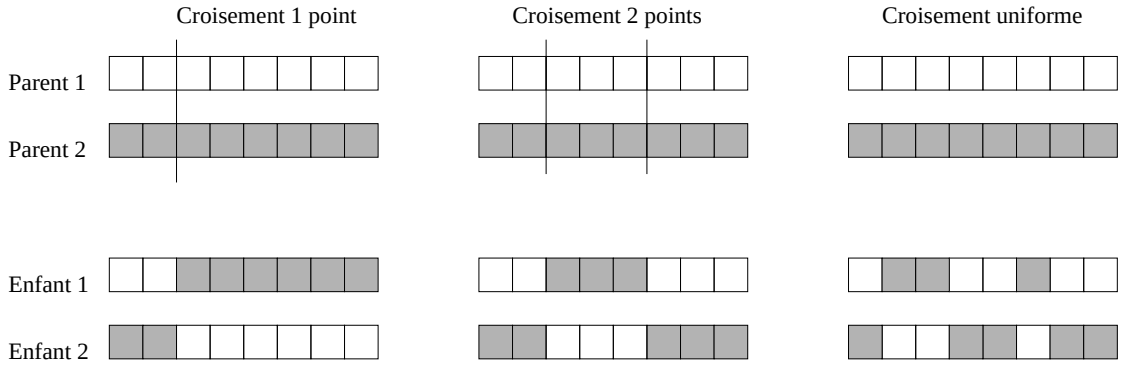


FIG. 2.7 – Exemples de reproduction grâce aux principaux types de croisement.

Il existe d'autres types de sélection moins utilisés dont on trouvera une description et une discussion dans [BFM00].

2.3.3.3 L'opérateur de croisement (crossover)

L'opérateur de croisement combine le matériel de un ou plusieurs parents pour obtenir un ou plusieurs enfants. Il existe différents types de croisement, nous allons brièvement présenter les trois principaux dont une illustration est donnée dans la figure 2.7. Le croisement un point détermine aléatoirement un point de coupure et échange la deuxième partie des deux parents. Le croisement deux points (qui peut être étendu à x points) possède 2 points (ou x) de coupures qui sont déterminés aléatoirement. Enfin le crossover uniforme échange chaque bit avec une probabilité fixé à $\frac{1}{2}$.

2.3.3.4 L'opérateur de mutation

Le rôle de la mutation consiste à faire apparaître de nouveaux gènes. Cet opérateur introduit une diversité nécessaire à l'exploration de l'espace de recherche en permettant de générer des

points dans des régions a priori sans intérêt [Ven96]. La mutation la plus simple sur un chromosome change un bit de façon aléatoire. Un chromosome a une probabilité de mutation d'un taux α (p_m).
Exemple :

1	0	1	0	0	1	⇒	1	0	0	0	0	1

2.3.3.5 Remplacement

Cette dernière étape du processus itératif consiste en l'incorporation des nouvelles solutions dans la population courante. Les nouvelles solutions sont ajoutées à la population courante en remplacement (total ou partiel) des anciennes solutions. Généralement, les meilleures solutions remplacent les plus mauvaises ; il en résulte une amélioration de la population. Lorsque la nouvelle population n'est constituée que de nouvelles solutions, on parle d'algorithme génétique générationnel.
Ceci peut se formaliser par l'utilisation de mécanismes appelés $\mu + \lambda$ ou (μ,λ) où μ correspond au nombre de parents utilisés pour générer λ enfants. Les μ parents de la génération suivante étant déterministiquement obtenu comme les meilleurs parmi les λ enfants (cas (μ,λ) ou les λ enfants plus les μ enfants (cas ($\mu + \lambda$)).

2.3.4 Conclusion

L'intérêt des algorithmes génétiques est de produire des solutions diversifiées notamment en utilisant des heuristiques telles les partages, en temps contrôlable (algorithme any time). Afin de les appliquer à un problème particulier, il est nécessaire de définir convenablement la fonction d'évaluation et le codage d'une solution. Ces algorithmes nécessitent également la définition d'opérateurs, aussi bien de mutation que de croisement, en adéquation avec le problème. Dans de nombreux cas, ils peuvent être difficiles à paramètrer et nous proposerons dans la suite de ce travail (voir chapitres 3 et 5) une méthode permettant de fixer adaptivement leurs taux d'application.

2.4 État de l'art sur les algorithmes génétiques développés pour l'extraction de connaissances

Les algorithmes génétiques ont été largement utilisés en extraction de connaissances pour chacune des différentes tâches. Les auteurs mettent en avant leurs performances dans le champ de l'optimisation et leur facilité d'utilisation et d'adaptation par rapport à des méthodes ad-hoc.
Nous allons présenter leur application à différentes tâches d'extraction de connaissances : la sélection d'attributs en classification, la sélection d'attributs en clustering, le clustering et la recherche de règles.

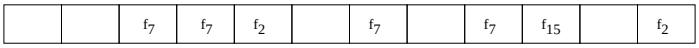


FIG. 2.8 – Exemple de génome proposé par Cherkauer représentant l'ensemble d'attribut sélectionné $\{f_2, f_7, f_15\}$.

2.4.1 Sélection d'attributs en classification

La sélection d'attributs consiste à rechercher un sous-ensemble d'attributs pour réduire le temps de traitement et/ou améliorer la classification. Il existe deux types de méthodes de sélection d'attributs : les méthodes enveloppantes (wrapper) qui se servent du classifieur pour évaluer le sous-ensemble d'attributs sélectionnés et les méthodes filtrantes où l'évaluation est faite indépendemment du classifieur. De plus amples détails sur cette tâche sont donnés dans le chapitre 4. La sélection d'attributs pour la classification est traitée dans la littérature aussi bien par des approches monocritères que multicritères.
Dans les approches monocritères, nous citerons les travaux les plus significatifs en nous attachant plus particulièrement à la représentation des solutions et les opérateurs mis en œuvre ainsi qu'aux fonctions d'évaluation utilisées [Fre02].
Il existe principalement deux types de codage des individus pour réaliser une sélection d'attributs à l'aide d'un algorithme génétique. La première représentation considère que l'espace de recherche peut être représenté par tous les sous-ensembles possibles d'attributs et code donc une solution par une chaîne de bits d'une longueur fixe de m bits où m représente le nombre d'attributs disponibles. Le $i^{ème}$ bit indique si le $i^{ème}$ attribut est sélectionné (valeur du bit à un) ou non (valeur du bit à zéro). L'avantage de ce codage réside dans sa simplicité et la possibilité d'utiliser des opérateurs de croisement et de mutation non dépendants du problème [VdJ93, YH98]. Certains auteurs proposent des opérateurs adaptés au problème de sélection d'attributs et notamment un opérateur de croisement, le "non-standard crossover" [CGSS99, GSCWS99, GSW99, GSW98]. En effet les auteurs considèrent que dans les opérateurs de croisement standards les zéros et les uns sont considérés de la même façon alors qu'ils n'apportent pas la même information. Dans [CGSS99], Chen considère que les uns qui représentent les attributs potentiellement intéressants apportent plus d'information que les zéros. L'opérateur de croisement utilisé est une forme de Hill-Climbing et permet, à partir des attributs sélectionnés communs, de générer de nouveaux individus.

La seconde représentation a été proposée par Cherkauer [CS96]. Dans cette représentation, un individu est de taille fixe égale au nombre d'attributs et est codé par l'index des attributs sélectionnés et des zéros (voir figure 2.8). L'auteur autorise les attributs à être présents plusieurs fois car il considère que cette redondance peut ralentir la perte de diversité. Pour lui permettre de travailler avec cette représentation, il utilise une variante du crossover uniforme et un opérateur de mutation. En plus de ces deux opérateurs, l'auteur introduit un opérateur "Delete Feature" qui, à partir d'un nouvel individu, génère un nouvel individu en lui enlevant un attribut sélectionné de façon aléatoire. Si l'individu possède cet attribut en plusieurs exemplaires tous sont supprimés.

L'évaluation d'un sous-ensemble d'attributs peut se faire soit par un classifieur, dont on évalue le taux d'erreur sur le sous-ensemble pour les méthodes enveloppantes, soit par une fonc-

tion spécifique pour les méthodes filtrantes où le classifieur n'est pas utilisé. Une grande majoritée de méthodes utilise les méthodes enveloppantes avec C4.5 [CS96], des tables de décision euclidiennes [GSW99, GSW98], des arbres de décision [VdJ93, BHV+95, CS96], les réseaux de neurones [OS94, OS95], les k plus proches voisins [PGPD95, PDPG93, RPG+97], ou d'autres [KJ97].
Dans certains cas, la fonction d'évaluation combine les performances du classifieur et d'autres mesures pour réaliser une sorte d'hybridation entre les méthodes enveloppantes et les méthodes filtrantes sous la forme d'une fonction d'évaluation multicritère traitée par aggrégation. Les multiples critères à optimiser incluent l'exactitude de la classification, le coût et le risque associés à une classification qui dépend de la sélection d'attributs utilisée pour décrire les données. Ainsi **GADistAl** est une approche multicritère utilisant un algorithme génétique associé à un algorithme d'apprentissage basé sur un réseau de neurones utilisant la distance inter-configuration [YH98].
La fonction d'évaluation bicritère de l'algorithme génétique est :

$$fitness(x) \;=\; accuracy(x) - \frac{cost(x)}{accuracy(x)+1} + cost_{max}$$

où :

- $accuracy(x)$ est le test de l'exactitude de la classification par le réseau de neurones utilisant DistAl pour le sous-ensemble d'attributs x,
- $cost(x)$ est la somme des mesures de coûts du sous-ensemble représenté par x,
- $cost_{max}$ est une borne supérieure sur les coûts des solutions candidates (dans [YH98] c'est la somme des coûts de tous les attributs).

Cette approche donne de très bons résultats. Elle permet en effet une réduction significative du nombre d'attributs et elle obtient un taux d'exactitude très élevé.
D'autres approches utilisent une fonction qui aggrège la qualité de la classification et d'autres mesures notamment dérivées de la théorie de l'information [BHV+95].
MOEA est une approche réellement multicritère développée par Emmanouillidis, Hunter et MacIntyre qui utilise les réseaux de neurones comme classifieur [EHM00].
Leurs critères d'évaluation sont, pour cet algorithme :
- le taux de mauvaise classification estimé pour le classifieur,
- le nombre d'attributs sélectionnés.

Une telle fonction multiobjective empêche de s'imposer une restriction a-priori sur l'espace de recherche. L'algorithme montre un pouvoir d'exploration à travers les solutions non-dominées. Enfin, la méthode semble être relativement générique pour être appliquée sur différents classifieurs et sur des problèmes de grande dimension.

Les approches filtrantes sont relativement moins nombreuses [BDH+96, CL99a]. Dans [BDH+96], l'auteur utilise une fonction d'évaluation qui prend en compte le nombre d'attributs et favorise un nombre peu élevé d'attributs sélectionnés. Cette fonction prend également en compte une fonction dérivée de la théorie de l'information.

La sélection d'attributs en classification par algorithmes génétiques a été proposée dans de nombreux travaux. La plupart des algorithmes de la littérature propose un codage simple binaire (attribut sélectionné ou non) et la majorité des algorithmes utilisent une méthode enveloppante. L'approche par méthode évolutionnaire se justifie relativement bien par rapport aux méthodes traditionnelles dédiées à cet usage [YH97]. En effet, les méthodes traditionnelles supposent que la

mesure de qualité utilisée pour évaluer un sous-ensemble d'attributs est monotone (par exemple pour les méthodes de type wrapper, la qualité de prédiction) or la monotonocité n'est pas toujours vérifiée si les attributs sont bruités ou corrélés. De plus, la plupart des méthodes sont inadaptables lorsque l'on désire considérer plusieurs critères pour évaluer le sous-ensemble d'attributs.

2.4.2 Sélection d'attributs pour le clustering

L'utilisation de la sélection d'attributs en clustering est moins répandue que son utilisation pour la classification.
Dans le cas du clustering la représentation habituelle de la sélection d'attributs vue pour la classification est toujours de même nature : un sous-ensemble d'attributs parmi ceux possibles.
Dans [KSM00], Y. Kim et al. proposent un algorithme évolutionnaire multiobjectif, **ELSA**, pour réaliser la sélection d'attributs dans un cadre d'apprentissage non-supervisé. Les objectifs sont :

a. F_{within} qui favorise les clusters denses en regardant la cohésion du cluster. Soient x_i, i=1,...,n les données, $x_{i,j}$ le $j^{\text{ème}}$ attribut de x_i, d la dimension du sous ensemble d'attributs J, K le nombre de clusters, α_{ik} l'appartenance ou non de x_i au cluster k et le centre du cluster γ_{kj}, Z_W constante de normalisation :

$$F_{within} = 1 - \frac{1}{Z_W}\frac{1}{d}\sum_{k=1}^{K}\sum_{i=1}^{n}\alpha_{ik}\sum_{j\in J}(x_{ij}-\gamma_{kj})^2$$

b. $F_{between}$ qui favorise des clusters bien séparés en mesurant leur distance au centre global avec Z_b une constante de normalisation :

$$F_{between} = \frac{1}{Z_b}\frac{1}{d}\frac{1}{k-1}\sum_{k=1}^{K}\sum_{i=1}^{n}(1-\alpha_{ik})\sum_{j\in J}(x_{ij}-\gamma_{kj})^2$$

c. $F_{cluster}$ qui indique que moins il y a de clusters plus le modèle est compréhensible avec $K_{max}(K_{min})$ le nombre de clusters maximum (minimum) que l'utilisateur désire obtenir.

$$F_{cluster} = 1 - \frac{K - K_{min}}{K_{max} - K_{min}}$$

d. $F_{complexite}$ qui favorise un nombre minimal d'attributs sélectionnés avec D le nombre total d'attributs, l'objectif étant d'avoir une complexité minimum pour avoir un modèle plus compréhensible et une meilleure généralisation :

$$F_{complexite} = 1 - \frac{d-1}{D-1}$$

On peut remarquer que F_{within} et $F_{between}$ sont des résultats produits par Kmeans ce qui caractérise une approche enveloppante (wrapper) alors que $F_{clusters}$ et $F_{complexite}$ sont indépendants du résultat de Kmeans ce qui caractérise plutôt une approche filtrante.
La modélisation de l'algorithme évolutionnaire ELSA est la suivante :

- Représentation d'un agent : $D + K_{max} - 2$ bits, avec les D bits qui correspondent aux attributs sélectionnés.
- Opérateurs : la mutation est le seul opérateur utilisé.

– Évaluation : utilisation du Kmeans standard pour construire les clusters donnés par les D bits d'un agent, avec K donné par l'agent.

Cet algorithme donne de bons résultats sur des bases de données possédant beaucoup d'exemples mais relativement peu d'attributs. Une amélioration de l'algorithme pourrait venir par une parallélisation du Kmeans, une implémentation d'un crossover et une étude approfondie du front Pareto.

2.4.3 Clustering

Il existe de nombreux algorithmes de clustering par algorithmes génétiques. Ceux-ci peuvent être classés de différentes manières : en fonction de la sortie de l'algorithme, en fonction du type de clustering réalisé, ... Dans cet état de l'art, nous avons pris parti de présenter différents algorithmes existants en fonction du type de résultat qu'ils produisent, ce qui peut s'apparenter au type de codage utilisé : une description des clusters, les coordonnées des centres des clusters ou l'appartenance des instances à un cluster. Nous présenterons également une approche différente des précédentes qui utilise les règles d'associations. Nous pouvons également trouver dans la littérature deux types d'approches relativement différentes qui, dans un cas, indique le nombre de clusters à trouver et dans un autre demande à l'algorithme de proposer un nombre de clusters optimal. Ce second type d'approche est beaucoup plus rare et dans la majorité des travaux, les auteurs fixent le nombre de clusters.

Une façon de résoudre le problème du clustering est de demander à l'algorithme génétique de rechercher la description du cluster. L'espace de recherche pouvant être très grand, la recherche se limite souvent à un seul type de forme de cluster. Ainsi dans [SGW$^+$95], les auteurs proposent de rechercher un nombre variable de clusters de forme ellipsoïde. Dans [GF96], Ghozeil et al. proposent quant à eux, de rechercher des clusters de forme hyperpavé (pavé de dimension pouvant être supérieure à 3).

Certains travaux proposent de résoudre le clustering en essayant de trouver les centres de cluster optimaux. Cette approche peut aussi bien s'appliquer à une approche de type médioïde où le centre est un objet à classer, qu'à une approche de type centroïde où le centre est juste représenté par des coordonnées ne correspondant pas obligatoirement à un objet. L'approche de clustering alors utilisée est proche du principe des Kmeans et consiste juste à répartir les instances en fonction de la distance au centre.

Dans [MB00, BM94], les auteurs proposent de coder un individu comme un centre initial du clustering Kmeans. Pour un espace à N dimensions et K clusters désirés, un individu est donc représenté par une chaîne de $N \times K$ mots. Dans le type d'approche de Babu, l'évaluation d'un individu est donnée par l'algorithme Kmeans en l'appliquant sur l'individu à évaluer [BM94]. Ceci peut alors rendre l'algorithme très gourmand en temps de calcul car l'algorithme Kmeans sera exécuté de l'ordre de $P \times G$ fois où P est le nombre d'individus de la population et G le nombre de générations effectuées. Or la complexité de l'algorithme Kmeans est en $O(nkl)$ avec n le nombre d'instances, k le nombre de clusters et l le nombre d'itérations nécessaires pour obtenir une solution stable.

Dans [HOB99], Hall et al. proposent quand à eux de coder un individu par une matrice de M

lignes et K colonnes où M représente les attributs et K le nombre de clusters à trouver. Ainsi, chaque ligne représente un centre et ses coordonnées. Dans ce cas, le codage de l'algorithme génétique est dans le domaine du continu. La fonction d'évaluation utilisée est relativement classique et minimise la distance intra-cluster. Le crossover appliqué est un croisement deux points réalisé sur chaque centre des parents.
Dans [ECM00, ECM97b], Estivil et al. proposent un algorithme génétique hybride et itératif pour le clustering en optimisant le critère de clustering des médioïdes. La fonction d'évaluation est la minimisation du critère $L_1(C) = \sum_{i=1}^{n} EUCLID(x_i, REP[x_i, X])$ où $REP[x_i, X]$ représente le point le plus proche de x_i dans X, ensemble des points. La distance utilisée ici est euclidienne mais peut aussi être celle de Minkowski.
Les auteurs identifient deux problèmes à surmonter :

- les algorithmes génétiques et le recuit simulé sont en général plus lents que les méthodes itératives et souvent ils sont moins bons en terme de qualité que la meilleure solution d'un algorithme multi-départ itératif utilisant le même effort de calcul.
- l'hybridation des algorithmes génétiques et des méthodes itératives est possible mais souvent délicate.

Les auteurs décomposent l'algorithme Kmeans en plusieurs étapes et les utilisent comme opérateurs de mutation. L'algorithme génétique utilisé est celui de Bianchi et Church [BC92] : BC-GA car cet algorithme est celui donnant les meilleurs résultats pour la fonction L_1. L'algorithme génétique est codé en chaîne d'entiers de longueur k, le nombre de clusters, et ces entiers prennent leur valeur dans [1,n] où n représente le nombre d'objets. Les opérateurs sont $R3$ (Random Respectful Recombinaison) et RAR_w amélioré au niveau temps de calcul. L'hybridation intervient dans l'opérateur de mutation où une itération du Kmeans est effectuée d'où le nom **H-BC-GA** (Hybrid BC-GA) [Rad92]. Dans [ECM00], les auteurs comparent les deux approches (avec et sans hybridation) en donnant à chacune le même nombre d'opérations CPU. Dans [ECM97b], les auteurs se comparent à un Hill-climbing qu'ils ont eux même implémenté [ECM97a] et à l'utilisation ou non d'opérateurs de recombinaison dédiés.
Il apparaît que l'approche par médioïde est plus robuste que l'approche par centroïde [KM95] ce qui peut intuitivement se comprendre car en statistique la médiane est plus robuste aux extremums que la moyenne.

Dans d'autres approches, le clustering est réalisé directement grâce à un codage d'affectation aux clusters des instances à classer. Ce codage a comme inconvénient d'avoir un fort taux de redondance, en effet les individus 1 2 2 1 et 2 1 1 2 représentent le même cluster mais cela, par exemple, offre une diversité intéressante pour le croisement.
Krishma et al. [KM99] ont ainsi proposé un algorithme génétique hybride reprenant cette représentation dans GKA (Genetic Kmeans Algorithm) en utilisant Kmeans comme opérateur (KMO). On peut observer que l'application de KMO est intéressante dans les premiers temps de la recherche mais moins après. Il serait donc intéressant de pouvoir mesurer finement son application en utilisant des techniques avancées (mécanismes adaptatifs par exemple).
Dans [LL99], Lozano et Larra proposent un algorithme génétique pour réaliser un clustering hiérarchique optimisant une ultramétrique distance qui définira la qualité de la classification hiérarchique. Étant donnée une matrice de dissimilarité $D = [d_{i,j}]_{i,j=1,2,\ldots,n}$ d'un ensemble de données l'objectif est de trouver la distance ultramétrique qui minimise $\sum_{i<j}(d_{i,j} - \delta i, j)^2$. Les auteurs ont développé un algorithme génétique basé sur l'ordre. Ils utilisent quatre opérateurs de crossover :

PMX, CX, OX1 et AP et six opérateurs de mutation SM, SIM, ISM, IVM, EM et DM [LKM+99]. Les auteurs comparent ces différents opérateurs ainsi que l'algorithme génétique avec un "branch and bound", un algorithme de programmation mathématique et la méthode de projection itérative. Les comparaisons s'effectuent par rapport au pourcentage de variances des dissimilarités qui est égal au coefficient de correlation cophonetic en prenant son maximum, son minimum et sa moyenne : $VAF = 1 - \frac{\sum_{i<j}(\delta_{i,j} - d_{i,j})^2}{\sum_{i<j}(\delta_{i,j} - \delta)}$.

Enfin, certains auteurs proposent de traiter le problème du clustering comme un problème de règles d'association particulier [LSW97, HKKM97]. De ce fait, ils ont traité la recherche de règles pour le clustering par des méthodes évolutionnaires et notamment des algorithmes génétiques. Dans [SZT02], Sarafis et al. proposent un algorithme génétique pour réaliser un clustering avec un nombre de clusters fixé : **RBCGA**. Chaque règle possède d gènes où chaque gène i correspond à un intervalle impliquant un attribut comportant une borne inférieur lb_i et une borne supérieure ub_i. Les règles sont composées de conjonctions de terme (type AND). La fonction fitness utilisée est flexible et prend en considération l'asymétrie d'un cluster, la densité de la règle et sa couverture, l'homogénéité de la règle et bien sûr la dissimilarité interclasse et la similarité intraclasse. Ces notions définies ci-dessous sont soit pour les premières relatives aux règles d'association, soit spécifiques au clustering pour les dernières.
Asymétrie de la règle ($a_{(R)}$) : permet une distribution uniforme des exemples en fonction du centre de gravité du cluster. Plus le centre de la règle et le centre de gravité du cluster sont proches plus les exemples sont distribués uniformément autour du centre de gravité (centre du cluster).
Densité d'une règle et couverture : la densité $d_{(R)}$ représente le nombre de règles par unité de volume (sachant que le volume est défini par les bornes de la règle). La couverture $cov_{(R)}$ représente le nombre d'exemples couverts par la règle relatif au nombre total d'exemples à classer.
Tous les concepts relatifs à la règle sont utilisés pour calculer le poids d'une règle R f_R :

$$f_R = a_{(R)} \times cov_{(R)} \times d_{(R)}$$

Homogénéité de la règle : l'objectif de cette fonction est d'identifier et de quantifier les discontinuités dans la distribution des exemples pour toutes les dimensions.
Recouvrement de règles : lors de l'évolution des individus, des règles recouvrant de mêmes exemples peuvent apparaître. Les auteurs utilisent alors un mécanisme de pénalisation des individus représentant des règles qui se recouvrent en modifiant le nombre d'exemples totals couverts par la règle.
Les auteurs se comparent à Kmeans. L'avantage de cette approche est l'élimination du bruit, la découverte de clusters qui ne sont pas de forme sphériques mais, en contrepartie, RBCGA est difficile à paramétrer et a surtout un gros problème de mise à l'échelle, au niveau temps de calcul, par rapport au nombre d'exemples traités.

A première vue, l'approche par algorithme évolutionnaire pour résoudre le problème du clustering semble être intéressante. En effet, l'intérêt des algorithmes génétique est qu'ils effectuent une recherche globale alors que de nombreux algorithmes spécifiques de la littérature ne sont que des recherches locales. Mais la plupart des auteurs se comparent uniquement avec Kmeans qui est reconnu pour se laisser piéger par les optima locaux et être de moins en moins performant à mesure que le nombre de clusters augmente.

2.4.4 Règles d'association et de classification

Dans les études menées sur les règles d'association et plus particulièrement sur les règles de décision par algorithme génétique, il existe deux approches principales de modélisation d'un individu : l'une est appelée *Michigan* [Hol75] et l'autre *Pittsburg* [Smi83]. Dans l'approche *Michigan*, chaque individu représente une règle alors que dans l'approche *Pittsburg* chaque individu représente un ensemble de règles et donc un concept de disjonction complet. Cette seconde représentation est coûteuse en mémoire. Il existe plusieurs algorithmes génétiques qui utilisent l'approche *Pittsburg* : GABIL [JSG93], GIL [Jan93] et HDPDCS [PGI97].
L'approche *Michigan*, de part sa nature et l'utilisation de fonctions fitness qui, en général, mesurent la performance d'une règle en dehors des autres découvertes, aurait, sans utilisation de mécanismes spécifiques, tendance à converger vers une même règle et non un ensemble de règles. Par exemple [Wei99], Weiss utilise le "sharing" pour éviter la convergence vers une règle unique dans le cadre de la prédiction d'évènements rares. Les exemples classiques d'algorithmes génétiques utilisant cette représentation sont COGIN [GS93] et REGAL [GN95].
Les travaux sur la recherche de connaissances par algorithme génétique est relativement ancienne et on peut distinguer deux voies de recherche différentes : la recherche de règles propositionnelles et la recherche de règles de logique de premier ordre. En effet, Smith en 1980 propose dans sa thèse [Smi80] LS-1 qui introduit une représentation structurée basée sur la sémantique du domaine. Dans [Ven93, Ven94], Venturini introduit SIA un système d'évolution possédant un codage non binaire et permettant d'utiliser des connaissances du domaine sous forme de hiérarchies sur les constantes symboliques. La fonction d'évaluation est basée sur la classe des exemples et prend en compte les exemples mal classés et bien classés en les pondérant avec des seuils fixés par l'utilisateur. Ce travail est repris et étendu dans SIA01 [AV96, Aug00].
GABIL [JSG93] introduit la description de concepts booléens qui sont représentés sous la forme d'un ensemble de règles propositionnelles en utilisant un codage binaire à taille variable pour pouvoir représenter des concepts disjonctifs et permettre l'évolution de règles de classification pouvant se chevaucher. Les opérateurs utilisés sont relativement simples, car ils opèrent directement sur des bits avec des probabilités inverses pour la mutation par exemple. L'opérateur de crossover est un peu plus évolué car il permet l'ajout ou la suppression de règles en obligeant les points de coupure à se situer sur le même attribut. La fonction d'évaluation f d'un individu I cherche à maximiser les exemples bien classés. Soit p le pourcentage d'exemples bien classés, la fonction d'évaluation est alors :

$$f(I) = p^2$$

L'utilisation des algorithmes génétiques pour extraire des règles comme nous l'avons vu dans la section 2.2.2 a de nombreux champs d'applications : le commerce [MSB01, dlIDRS96], la biologie [SWBP01], ...

2.5 Conclusion

Dans ce chapitre, nous avons décrit les méthodes d'optimisation basées sur les métaheuristiques en les divisant en deux classes : les méthodes à solution unique et les méthodes à population de solutions. Nous avons pu constater au fur et à mesure d'un court état de l'art pour chaque méthode, que leur utilisation en extraction de connaissances est relativement répandue pour les

différentes tâches que nous désirons étudier : sélection d'attributs, clustering, règles d'association. Dans la majorité des études citées dans ce chapitre, les auteurs s'attardent sur la fonction d'évaluation d'une solution qui représente, en effet, souvent, la partie la plus sensible de ces méthodes mais aussi sur le codage qui va directement influer, comme nous l'avons vu dans ce chapitre, sur le type de résultats attendus. Cette première réflexion amène automatiquement à une étape d'étude et de conception d'opérateurs spécifiques.
Ce chapitre contient aussi une présentation générale des algorithmes génétiques qui va nous servir de support au long de cette thèse pour présenter les différents travaux que nous avons réalisés.

Chapitre 3

Problématiques étudiées

Les données traitées dans notre étude proviennent de problématiques réelles issues d'expérimentations du Laboratoire de Génétique des Maladies Multifactorielles (LGMM) de l'Institut de Biologie de Lille (IBL) (Institut Pasteur de Lille). Ce laboratoire travaille principalement sur deux maladies multifactorielles qui sont l'obésité et le diabète de type II (non insulino-dépendant). La mise en jeu de voies de régulations multiples dans les différents organes atteints par la maladie rend difficile une approche simple pour avancer dans la compréhension des mécanismes impliqués dans le développement de ce type de maladie. C'est dans ce contexte que l'approche génétique de ces maladies multifactorielles peut être considérée comme un moyen pour mieux comprendre certaines voies physiopathologiques impliquées. Elle permet notamment :

- de préciser la contribution au développement de l'obésité d'une protéine impliquée dans une voie de régulation donnée,
- d'analyser les interactions gènes-environnement.

Le but de ces études génétiques est de pouvoir développer, dans l'avenir, des traitements qui agissent sur les cibles mises en évidence grâce à la génétique.
Dans le cadre de ce partenariat, nous avons abordé l'étude génétique de ces maladies par deux méthodes différentes : l'étude de liaisons génétiques fondée sur la co-ségrégation de la maladie et des allèles à un locus marqueur au cours des générations et l'étude du déséquilibre de liaison entre des sujets non apparentés respectivement atteints et non atteints.
Nous présenterons, tout d'abord, les deux maladies auxquelles nous nous intéressons. Nous introduirons ensuite les différentes problématiques génétiques et le vocabulaire nécessaire à leur compréhension (rappelé en annexe dans le glossaire). Puis nous formaliserons les objectifs que nous nous fixons.

3.1 Maladies étudiées

Le LGMM s'intéresse aux maladies multifactorielles qui sont sous la dépendance de plusieurs facteurs génétiques et d'environnement. La composante génétique est dite, de ce fait, polygénique. Chaque gène a théoriquement un petit effet sur la susceptibilité à la maladie. Cependant, la situation est plus complexe :

- une même maladie, par exemple le cancer du sein ou la démence d'Alzheimer, peut être

monofactorielle dans un petit nombre de cas (5 à 10 % des cas) mais est multifactorielle dans la majorité des cas ;
- parmi les maladies multifactorielles, l'importance de la composante polygénique (c'est-à-dire le poids des facteurs génétiques) peut être plus ou moins importante ;
- la composante polygénique peut, dans certaines maladies, être oligogénique : plusieurs gènes ayant des effets majeurs détectables.

Les données auxquelles nous avons accès concernent l'obésité et le diabète de type II, maladies que nous présentons dans la section suivante.

3.1.1 Obésité

L'obésité est une maladie fréquente dont l'incidence est en augmentation constante dans les pays occidentaux. Elle représente un problème de santé publique compte tenu de sa mortalité et de sa morbidité. L'obésité est associée à de nombreuses complications : respiratoires (syndrome de l'apnée du sommeil), métaboliques (diabète non-insulino-dépendant, dyslipidémies...), cardio-vasculaires (athérosclérose, hypertension artérielle), rhumatologiques, rénales ou encore biliaires. La prévalence des cancers est également majorée. En dehors de ces répercutions, la qualité de vie des patients obèses sur le plan familial, social et professionnel est sérieusement altérée.
L'obésité touche 7 à 23% de la population européenne et 33% des américains. On estime que 8% des français adultes sont obèses. Le nombre de malades augmente considérablement dans les pays industrialisés et la maladie atteint des personnes de plus en plus jeunes. La prévalence de l'obésité chez l'enfant est en augmentation croissante.
L'obésité est définie comme un excès de masse grasse, reflétée en pratique clinique par l'indice de masse corporelle (IMC) ou Body Mass Index (BMI). Le BMI est le rapport du poids en kg sur la taille en mètre au carré.

$$BMI = \frac{P(kg)}{T(m)^2}$$

Pour des individus ayant un :
- BMI < 27 kg/m^2 : pas de surpoids
- $27 \leq$ BMI < 30 kg/m^2 : surpoids
- $30 \leq$ BMI < 40 kg/m^2 : obésité
- BMI $\geq$ 40 kg/m^2 : obésité massive

L'obésité provient d'un déséquilibre chronique entre les apports et les dépenses énergétiques conduisant à une balance énergétique positive. Des facteurs environnementaux (stress, industrialisation), métaboliques (âge, diminution de l'oxydation des acides gras, diabète), comportementaux (mode et type d'alimentation, activité physique) et génétiques contribuent à ce déséquilibre. C'est pourquoi l'obésité est une maladie complexe et multifactorielle.
Au plan tissulaire, plusieurs organes et voies métaboliques sont impliqués dans la physiopathologie de cette maladie. De façon schématique, l'obésité peut être considérée comme un dysfonctionnement du tissu adipeux (organe de stockage d'énergie) et de ses relations avec les nombreux organes impliqués dans la mise en réserve des nutriments et l'utilisation d'énergie : le système nerveux central, le muscle, le foie, le pancréas et le système digestif. Des altérations primaires, éventuellement génétiques, des capacités de stockage cellulaire et/ou des voies de régulation de ces organes paraissent jouer un rôle majeur, dans la prédisposition à l'obésité et à ses complications associées.

3.1.2 Diabète

Le diabète de type II concerne environ 2% de la population française et se mesure par une glycémie à jeun supérieur à 1,26 g/ l (7 mmoles/l). Il résulte à la fois d'un déficit de l'insulino-sécrétion et d'une insulino-résistance. Il est associé à une obésité dans 80% des cas. Il est le plus souvent polygénique résultant de l'association d'une prédisposition génétique et de facteurs environnementaux, en particulier le surpoids, la sédentarité, plus accessoirement la nature des glucides et des lipides de l'alimentation.
L'insulino-déficience responsable de l'hyperglycémie du diabète de type II est précédée par 10 ou 20 ans, d'hypersécrétion insulinique (hyperinsulinisme) secondaire à une insulino-résistance des tissus périphériques. L'anomalie métabolique fondamentale qui précède le diabète de type II est l'insulinorésistance.
Le diabète de type II est une maladie à prédisposition génétique ; en effet 50% des malades ont des antécédents familiaux de diabète, et chez les jumeaux monozygotes (vrais jumeaux), si l'un développe la maladie, l'autre a 100% de risque d'en être aussi atteint. C'est une maladie généralement polygénique et rarement monogénique.
L'obésité de type androïde dont l'indicateur est le rapport taille/hanche (Waist to Hip Ratio=WHR) supérieur à 0,95(Homme) ou 0,85 (Femme) est un facteur essentiel de l'apparition du diabète de type II. 80 % des malades sont obèses (*BMI* supérieur à $30 Kg/m^2$).
Le diabète de type II sans excès de poids correspond à des diabètes dits hétérogènes généralement secondaires (hémochromatose, pancréatite, ...). Le risque de voir se développer un diabète de type II augmente avec l'âge notamment après 40 ans.

3.2 Études familiales par la méthode de paires de germain

Avant de présenter cette étude sur la méthode des paires de germain ("Identical By Descent"), nous introduisons quelques définitions de biologie nécessaires à sa compréhension.

3.2.1 Définitions de biologie

Définition 1 *Un locus représente un endroit précis du génome. On fait en général l'hypothèse qu'il s'agit d'un segment assez petit pour qu'il n'y ait pas de recombinaisons*[1] *internes.*

Définition 2 *Un allèle est une version possible d'un locus, il est donc constitué d'un enchaînement de nucléotides (il s'agit d'un fragment d'ADN). Chez un individu, chaque gène est représenté par deux allèles, situés au même locus (emplacement du gène sur le chromosome). Lorsque ces deux allèles sont identiques dans leur composition nucléotidique, l'individu est alors homozygote pour ce gène, s'ils sont différents dans leur composition, l'individu est alors hétérozygote pour ce gène.*

[1] Recombinaison : échange physique de portions de chromosomes issus des gamètes parentaux lors de la méiose, entrainant un réarrangement des combinaisons génétiques dans les descendants.

Définition 3 *Les marqueurs sont des fragments d'ADN qui permettent de caractériser un individu donné car ils présentent plusieurs versions possibles des allèles. De ce fait, ils peuvent servir de repères pour suivre la transmission d'un segment de chromosome d'une génération à l'autre. Ils sont notamment utilisés pour rechercher les gènes qui gouvernent certains caractères biologiques que l'on cherche à étudier. Le séquençage de ces marqueurs à un locus donné*[2] *(emplacement sur le génome) peut montrer certaines variations. L'existence de ces différentes formes définit ce qu'on appelle le polymorphisme génétique.*

Définition 4 *Méïose (à ne pas confondre avec mitose) : la méïose est le processus essentiel dans la reproduction humaine. La méïose est le processus cellulaire qui produit les gamètes, ovule ou spermatozoïde qui se combinent à la reproduction pour devenir un nouvel individu. Durant la méïose, une cellule avec une paire de chromosomes (cellule diploïde) est divisé en quatre cellules avec une chromatide dans chacune (quatre cellules haploïdes). (Voir figure 3.1.)*

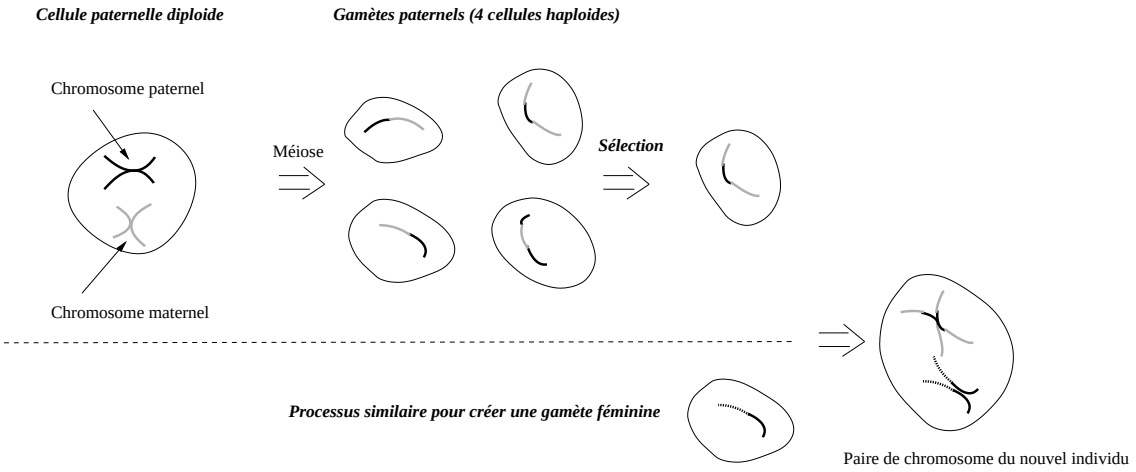


FIG. 3.1 – Illustration de la méiose sur une paire de chromosomes humains.

Définition 5 *Génotype / Phénotype : le génotype est constitué par l'ensemble des caractères héréditaires propres à un individu. La combinaison des deux gènes situés face à face sur les deux chromosomes homologues s'appelle le génotype.*
Le phénotype correspond à l'expression et à la réalisation de ce patrimoine génétique dans un environnement donné. Il rend compte des caractéristiques anatomiques et physiologiques d'un individu. L'existence de gènes dominants et récessifs explique qu'à un même phénotype correspondent des génotypes différents.

Chaque chromosome (cf. Figure 3.2) possède donc, pour coder un même gène, deux allèles : l'un provenant de la mère, l'autre provenant du père.

[2]Pour chaque marqueur, on connaît son emplacement sur la carte génétique.

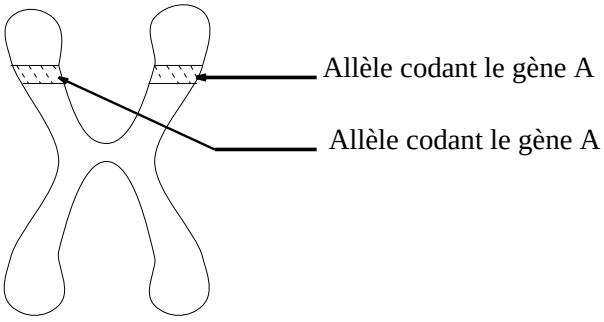


FIG. 3.2 – Exemple de chromosome.

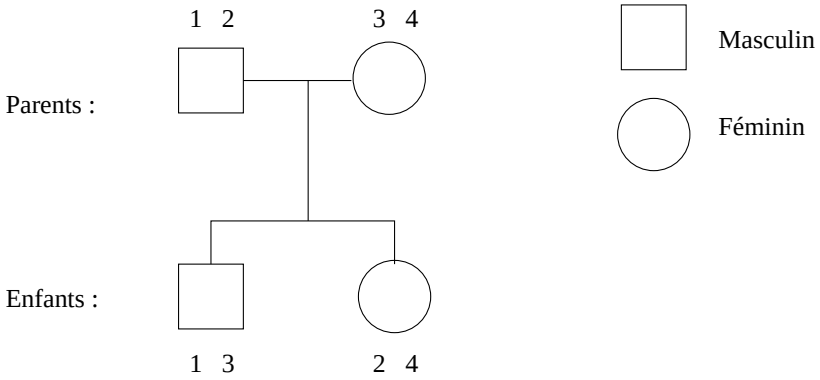


FIG. 3.3 – Exemple 1 de famille complètement informative.

Dans le cas d'une famille complètement informative de deux parents et deux enfants (cf. Figure 3.3), nous savons avec certitude quel allèle (1, 2, 3 ou 4) provient de quel parent et donc nous connaissons exactement la ressemblance génotypique entre les deux enfants ou entre parent et enfant.
Lorsque le cas de la figure 3.4 se présente, nous ne pouvons pas être certain que l'allèle 1 de l'enfant 1 soit hérité du père ou de la mère. On parle alors de probabilité car il existe une incertitude sur la provenance de l'allèle. Cette notion de probabilité intervient aussi lorsqu'il y a des données manquantes (par exemple l'un des deux parents est mort et nous ne connaissons pas son génotype).

Définition 6 *Le Morgan M est un taux de recombinaison pour une méiose. Le centi-Morgan cM représente 1 recombinaison sur 100 méioses observées. Le cM est une unité de distance en cartographie génétique.*

La fréquence de recombinaison de deux gènes liés (situés sur un même chromosome) est fonction de la distance qui les sépare. Lorsque les deux gènes sont proches, la probabilité de croisement est moindre que lorsqu'ils sont éloignés. Une évaluation du nombre possible de nouvelles recombinaisons se calcule en comptant le nombre de chiasma (croisement) durant la méiose. L'index de recombinaison se calcule en fonction du nombre de croisements décelés dans la même cellule au stade de la méiose. Le nombre de chiasma est à peu près constant d'une espèce à une autre,

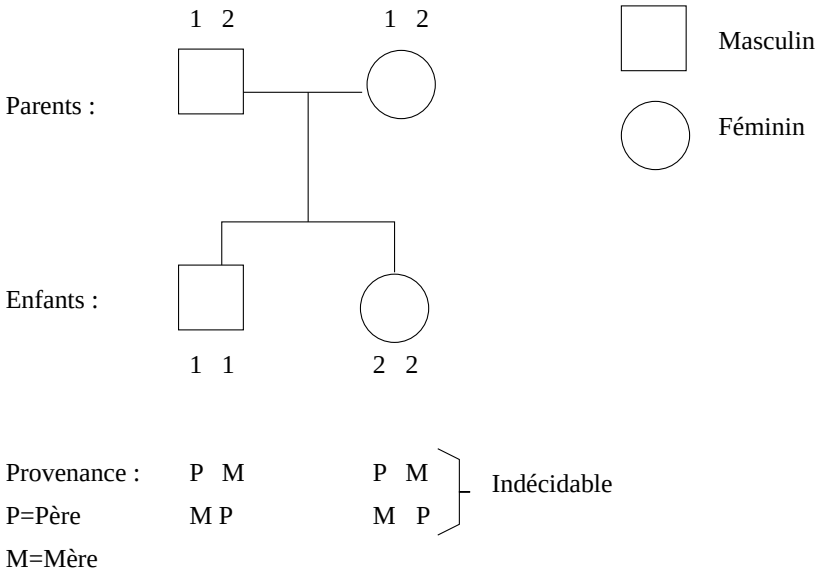


FIG. 3.4 – Exemple 2 de famille : cas avec ambiguité.

mais certains facteurs tels que la température, les radiations, les produits chimiques et la nutrition peuvent le modifier.

3.2.2 Problématique des études familiales ou de liaison

Les études familiales ou de liaison consistent à rechercher la présence d'une co-ségrégation entre le locus morbide et la maladie, c'est à dire la transmission conjointe, à travers les générations, d'un polymorphisme génétique et du trait pathologique dans des familles comportant plusieurs sujets atteints. Cette co-ségrégation à travers les générations est recherchée au sein de familles porteuses de l'affection, par l'analyse des différentes méioses et du fait des phénomènes de recombinaison. Elle traduit une liaison génétique permettant d'impliquer une région particulière du génome dans la détermination de la maladie. Une liaison marqueur-maladie ne signifie pas pour autant une liaison entre un allèle donné et la maladie. Afin d'étudier cette co-ségrégation de marqueurs, des méthodes d'analyses non paramétriques qualitatives et quantitatives sont utilisées. Les méthodes non paramétriques ne nécessitent pas de connaître le mode de transmission de la maladie, ni la pénétrance ou la prévalence des gènes associés.
La méthode des paires germains (frères et soeurs) atteints (affected sib pairs) est une méthode d'analyse de liaison "indépendante du modèle". Son but est de montrer que la transmission d'un segment chromosomique, dans des paires de sujets atteints provenant de fratries, n'est pas compatible avec une ségrégation aléatoire, mais qu'au contraire, ces sujets héritent d'un même segment chromosomique parental plus souvent que ne le voudrait le hasard. Pour un locus marqueur donné, elle cherche à estimer, au sein de paires de patients, la proportion d'allèles identiques par descendance (identical by descent, IBD), c'est à dire provenant d'un même allèle parental. Si le marqueur ségrège indépendamment de la maladie, la proportion de paires de germains atteints partageant 0, 1 ou 2 allèles IBD au locus marqueur, est respectivement 1/4, 1/2, 1/4. Si ce locus marqueur est lié

au locus de la maladie, deux frères atteints auront reçu plus de 50% d'allèles IBD.

3.2.3 Etat de l'art

Les études de liaison pour la détection de gènes impliqués dans les maladies sont relativement nombreuses dans la littérature. De nombreuses études portent sur la découverte de l'implication d'un seul gène (ou plus souvent marqueur) dans les maladies complexes par calcul de "LOD score" [BAB+96, KN97] ou d'un χ^2. Dans certaines études, les auteurs recherchent certaines interactions entre les gènes et les facteurs environnementaux pour des maladies telle que la maladie de Chron [LS94, HC01] ou des maladies de type psychiatrique [CB01]. La majorité de ces études utilisent des méthodes basées sur les statistiques pour découvrir des associations et réaliser le modèle génétique en utilisant les chaînes de Markov [LG87], les réseaux bayesiens [Gud] ou encore des modèles mathématiques comme les transformées de Fourier [KL98].

3.2.4 Données

A cause des données manquantes (par exemple parents décédés) ou non informatives (par exemple individu homozygote, i.e. ayant deux allèles identiques au même locus) on ne peut pas toujours déterminer avec certitude le nombre d'allèles IBD. On travaillera donc avec trois probabilités :

- $Z0$: probabilité de partager 0 allèle IBD
- $Z1$: probabilité de partager 1 allèle IBD
- $Z2$: probabilité de partager 2 allèles IBD

Les données disponibles sont donc :

- Pour chaque paire, les trois probabilités Z0, Z1 et Z2, calculées en 400 points (marqueurs ou loci), répartis régulièrement sur l'ensemble des chromosomes.
- Pour chaque paire, des covariables (environ une dizaine) qui dénotent le partage par la paire d'individus d'une caractéristique (éventuellement quantitative).
  Exemple : obésité, classe d'âge ...

Les données du LGMM nous sont fournies sous forme de plusieurs fichiers :

- un fichier donnant la liste des marqueurs,
- un fichier sous la forme donnée par le tableau 3.1 indiquant, pour chaque paire p_i d'individus d'un pédigré donné (famille) participant à l'étude, le partage de facteurs environnementaux (Affection *Aff*) et des loci,
- un fichier de description pour les paires indiquant combien il y a de paires par familles concernées,
- un fichier par chromosome donnant la liste des marqueurs (ou locus) et leur distance

Dans un premier temps, nous allons essayer de connaître les loci impliqués dans la maladie en fonction des données génotypiques.
Nous ne nous occuperons que des paires concernant des individus tous deux atteints par la maladie. En effet, le statut de non atteint est difficile à confirmer car chaque individu non atteint peut devenir atteint. Cela représente environ 200 paires.

Paire / Position	**Aff I1**	**Aff I2**	**locus 1**	**...**	**locus n**
p_1 pédigré, I1, I2	0, 1 ou 2	0, 1 ou 2	$Z0_{1,1}\ Z1_{1,1}\ Z2_{1,1}$	...	$Z0_{1,n}\ Z1_{1,n}\ Z2_{1,n}$
...	...	...	...	...	...
p_m pédigré, I1, I2	0, 1 ou 2	0, 1 ou 2	$Z0_{m,1}\ Z1_{m,1}\ Z2_{m,1}$	...	$Z0_{m,n}\ Z1_{m,n}\ Z2_{m,n}$

TAB. 3.1 – Fichier indiquant pour chaque paire les partages en chacun des loci.

Pour chaque paire sont indiqués le pédigré (la famille) et le numéro des individus du pédigré. Pour chaque individu de chaque paire, la valeur du facteur (Aff) est indiquée.

Pour pouvoir travailler avec des valeurs binaires, nous introduisons la notion de seuil pour mettre l'IBD à 0 lorsqu'on est en dessous d'un seuil déterminé par les experts et à 1 sinon.
Il existe plusieurs façons de représenter l'information :

1. Prendre chacune des probabilités Z0, Z1 et Z2 :
   - $Z0$ (0 allèle en commun pour le marqueur) : seuil égal à 0.25 (0.25 représente en effet la probabilité pour deux frères de n'avoir aucun allèle en commun).
   - $Z1$ (1 allèle en commun pour le marqueur) : seuil égal à 0.50 (0.5 représente la probabilité d'avoir un allèle en commun).
   - $Z2$ (2 allèles en commun pour le marqueur) : seuil égal à 0.25 (0.25 représente la probabilité d'avoir deux allèles en commun).
2. Prendre uniquement $Z0$ avec un seuil de 0.25 et considérer de la même façon ceux qui ont 1 ou 2 allèles en commun.
3. Calculer le π_{moyen}.

La notion de π_{moyen} nous donne la proportion moyenne de ressemblance.
Pour chaque marqueur k d'un couple d'individus i, nous avons :
$\pi_{i,k} = Z0_{i,k\ Observé} \times 0 + Z1_{i,k\ Observé} \times 0.5 + Z2_{i,k\ Observé} \times 1.$
Le $\pi_{moyen\ attendu}$ pour un couple peut être calculé comme suit :
$\pi_{moyen\ attendu} = Z0_{attendu} \times 0 + Z1_{attendu} \times 0.5 + Z2_{attendu} \times 1$
avec par exemple si la paire d'individus concerne deux frères :
$Z0_{attendu}$ = 0.25
$Z1_{attendu}$ = 0.50
$Z2_{attendu}$ = 0.25
$\pi_{moyen\ attendu}$ = 0.50
Ensuite, on pourra rendre binaire l'information obtenue en appliquant la notion de seuil.
Exemple :
Si $\pi_{moyen\ attendu}$ - $\pi_{moyen\ observé}$ > seuil, la valeur binaire de l'IBD sera 1,
sinon la valeur binaire de l'IBD sera de 0.

Les IBD ont été calculées en multi-points par GeneHunter, un logiciel d'analyse de liaison utilisé par le LGMM (cf http ://www.qpsf.edu.au/software/genehunter.html). Les valeurs des IBD ont été ensuite extraites à chaque intervalle de 5 cM.

Données académiques : Nous allons également travailler sur des jeux de données, dont on connaît les résultats, issus du Workshop GAW11 (cf. Annexe B) organisé par le Dr David A. Greenberg[3] en 1998. Cela va nous permettre de tester nos algorithmes et la validité de notre classification.

3.2.5 Objectif

Le LGMM disposant de ces données génétiques voudrait connaître les interactions gène-gène et gène-environnement pour une maladie donnée. Pour le moment, le LGMM ne dispose que d'outils statistiques permettant au plus la détection d'une interaction mais elle voudrait pouvoir mieux les détecter. Pour étudier ce type de maladies dites maladies multi-factorielles, le LGMM a besoin d'un outil d'analyse.

Nous désirons connaître parmi l'ensemble des facteurs (génétiques ou environnementaux) les sous-ensembles de facteurs liés à la maladie étudiée. Dans notre étude, les sous-ensembles contiendront différents loci et covariables d'environnement (nous désirons en effet connaître l'interaction gène-gène et gène-environnement).
Une hypothèse formulée par les biologistes est qu'il n'existe pas un seul groupe de facteurs de prédisposition mais plusieurs pouvant contenir des facteurs communs. De plus, une particularité de cette problématique est que le statut des non-malades sous étude n'est pas certain. En effet, une personne non malade a un instant t peut le devenir plus tard. Nous nous plaçons dans un contexte de classification non supervisé où nous recherchons à regrouper les individus partageant les même caractéristiques.

Dans un premier temps, nous voulons restreindre le nombre d'attributs (les marqueurs et les covariables) pour ne garder que les attributs pertinents afin de déterminer les attributs intervenant dans le fait que le patient soit atteint d'une maladie.
Nous désirons réaliser pour cela une sélection d'attributs ("Feature Selection"). Réduire l'ensemble des attributs considérés peut augmenter la précision de la prédiction ou la vitesse de traitement des données. Le problème de sélection d'attributs implique de trouver "un bon ensemble" d'attributs sous une certaine fonction objective. En d'autres termes, le but est de trouver un ensemble d'attributs significatifs qui, présenté au classifieur, maximise sa performance.
Ensuite, nous désirons réaliser à partir de ces attributs pertinents, une classification à l'aide d'un algorithme de clustering. Cela nous permettra de regrouper les paires d'individus en fonction de leur ressemblance.

3.2.6 Création de benchmarks

Les études que nous menons sur les maladies multifactorielles nous conduisent à traiter des problèmes possédant des caractéristiques particulières. Nous avons décidé de mettre à disposition

[3]Dr David A ; Greenberg, Department of Psychiatry, Box 1229, Mt Sinai Medical center, 1 Gustave Levy Place, New York 10029

un simulateur qui permet de générer des benchmarks similaires à notre problématique d'étude familiale sur le diabète et plus particulièrement sur des paires d'individus. Ce simulateur prend en effet en compte :

- la corrélation entre les gènes en fonction d'une distance de corrélation,
- la présence de coupures chromosomiques,
- plusieurs associations de plusieurs gènes à découvrir.

Objectif : Ces benchmarks vont nous permettre de simuler la problématique de façon plus fidèle que le premier jeu de données que nous avions à disposition. De plus, nous pourrons tester nos approches sur des jeux de données dont nous connaissons les particularités. Ainsi si il y avait au

Algorithme 4 Pseudo code du générateur

Procédure : propagation(individu,chromosome,marqueur de début,σ)
Variables locales : $t[][]$ matrice des marqueurs impliqués par association, $allele[]$ tableau des valeurs de partage d'un individu
Paramètres : Nombre de paires d'individus atteints nb_paires, Nombre de chromosomes nb_chrom, nombre de marqueurs par chromosome $nb\_marqueurs$, nombre d'associations, Pénétrance des associations, Nombre minimum de marqueurs par association, Nombre maximal de marqueurs par association, taux de recombinaison σ

```
for (n = 0 to nb_paires) do
  for (i = 0 to nb_chrom) do
    if (∃ association sur i) then
      for (j = 0 to nb_marqueurs) do
        if (∃ association sur le marqueur j) then
          n.allele[j] = 2
        end if
      end for
    end if
    propagation(n,i,nb_marqueurs,σ)
  end for
end for
```

marqueur n un partage de 2 allèles entre la paire d'individus pour le marqueur n+1 il y a plusieurs possibilités :

- il n'y pas eu recombinaison : la paire partage toujours 2 allèles,
- il y a eu une seule recombinaison méïotique : la paire partage alors 1 seul allèle,
- il y a eu plusieurs recombinaisons méïotiques : la probabilité pour passer d'un partage à l'autre de 1 vers 2 ou de 1 vers 0 est calculée à chaque recombinaison,
- le marqueur n+1 a déjà une valeur obtenue par propagation du partage d'un marqueur situé à sa droite.

3.2.6.1 Mise en œuvre

Le benchmark doit simuler le partage d'allèles entre les paires d'individus. Une paire peut partager 0, 1 ou 2 allèles par marqueur. Un marqueur faisant parti d'une association provoquant la maladie induit un partage de 2 allèles pour la paire d'individus. L'algorithme (voir algorithme 4 et

Algorithme 5 Pseudo code du générateur : fonction de propogation

Procédure : α permet de donner la valeur de partage en fonction du taux de recombinaison et du voisin de référence
Paramètres : Individu n, Chromosome c, Nombre de marqueurs m, σ taux de recombinaison
fini=FALSE, i=1
while non fini **do**
  fini=TRUE
  for (∀ j | n.allele[j]==2) **do**
    /*Chaque point de l'assocation*/
    /*propagation à gauche et à droite*/
    if (j+i) non marqué **then**
      /*On a jamais affecté de valeur à j+i*/
      $n.allele[j+i]$=$\alpha(\sigma, j+i-1)$
      marquer(j+i)
      fini=FALSE
    end if
    if (j-i) non marqué **then**
      $n.allele[j-i]$=$\alpha(\sigma, j-i+1)$
      marquer(j-i)
      fini=FALSE
    end if
  end for
  i++
end while

algorithme 5) part des points d'association situés sur un même chromosome et propage simultanément le partage allélique. En effet, si un marqueur est partagé, ses voisins ont une forte probabilité de l'être. Pour cela, l'algorithme teste pour chaque marqueur rencontré le taux de recombinaison pour savoir s'il y a eu recombinaison méïotique pour ce marqueur par rapport au marqueur traité précédemment. L'algorithme continue la propogation jusqu'à ce qu'il rencontre soit une coupure chromosomique soit une autre propagation. Il traite alors un nouveau chromosome.

La figure 3.5 illustre le procédé de propagation du partage allélique.

3.2.6.2 Description des fichiers

Le programme crée plusieurs fichiers à partir d'un nom de benchmark nom que l'on lui donne :
- nom_affected.dat
- nom_descr.dat contient un descriptif des paramètres utilisés pour créer le fichier de benchmark
  - paramètres donnés par l'utilisateur :
    - Le nombre de paires d'individus affectés,
    - Le nombre de chromosomes,
    - Le nombre de marqueurs,

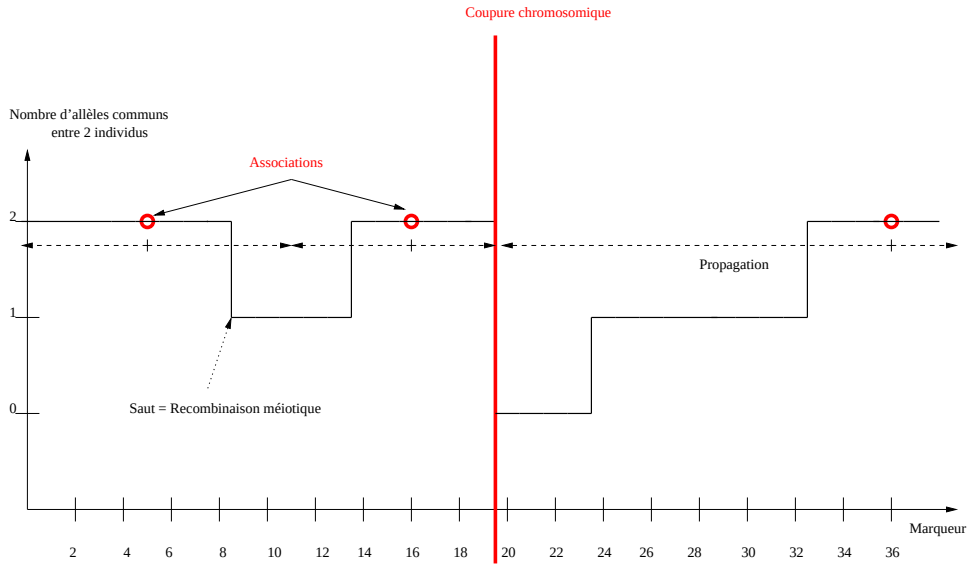


FIG. 3.5 – Exemple pour une paire d'individus de partage allélique et des influences des associations de marqueurs.

  - La distance de recombinaison,
  - Le nombre d'associations désirées,
- paramètres générés par le programme :
  - Les associations et les marqueurs impliqués,

Exemple

Nous donnons ici un exemple de fichiers générés par notre générateur de benchmark pour les données d'entrée suivantes :

```
Nombre d'individus : 20
Nombre de chromosomes : 5
Nombre de marqueurs : 40
Distance de recombination : 4
Nombre d'associations : 2
Association 1 : 33
Association 1 : 135

Association 2 : 162
Association 2 : 197
```

Les figures 3.6 et 3.7 présentent pour deux paires d'individus un exemple de partage d'allèles obtenu à l'aide du générateur de benchmark.

La figure 3.8 présente pour les 20 paires d'individus considérées, la moyenne des partages allè-

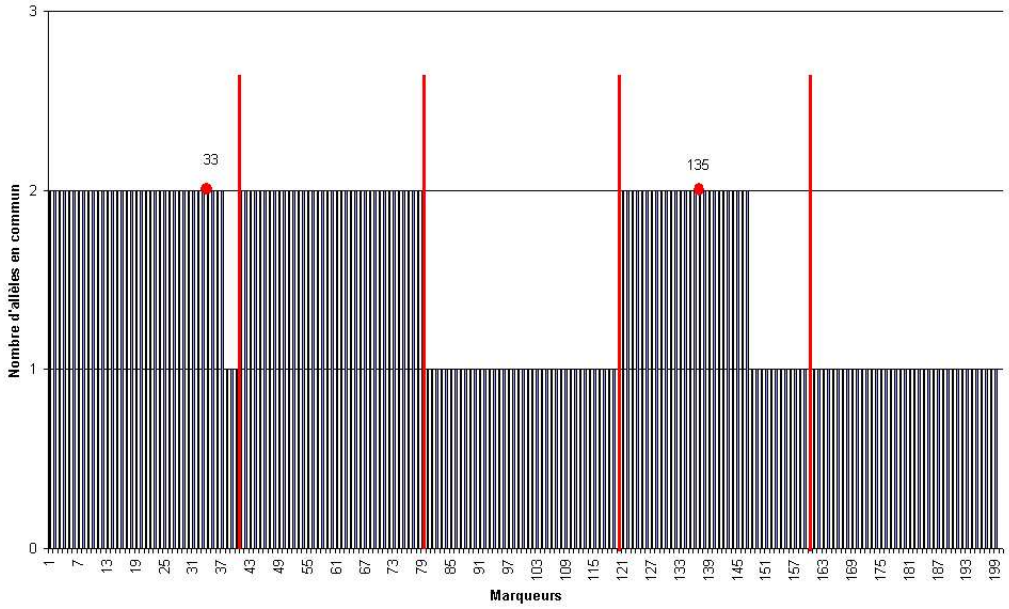


FIG. 3.6 – Exemple pour une paire d'individus de partage allélique et des influences des associations de marqueurs (33 et 135) obtenues par le générateur de benchmark.

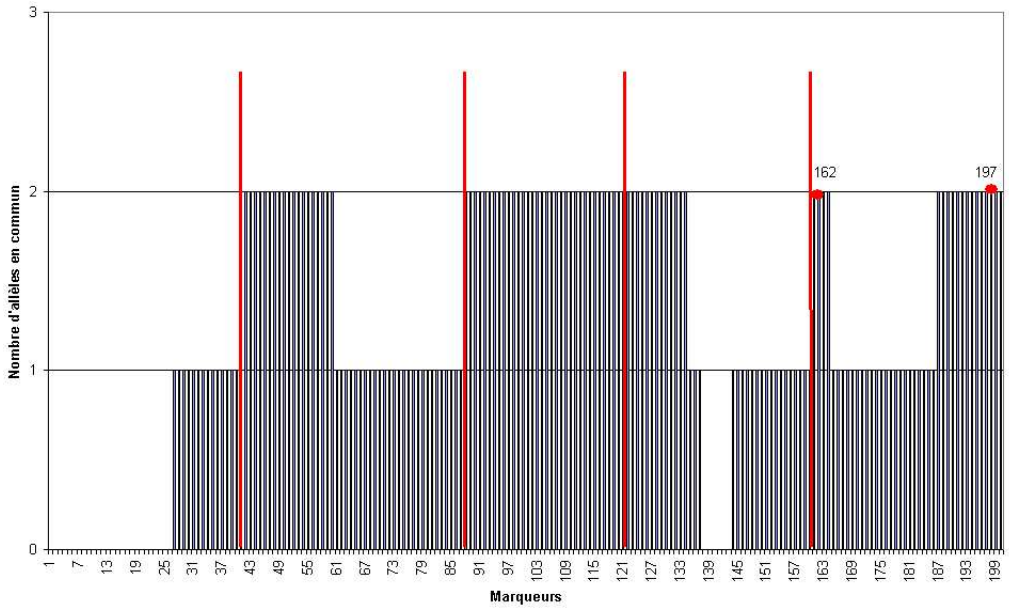


FIG. 3.7 – Exemple pour une paire d'individus de partage allélique et des influences des associations de marqueurs (162 et 197) obtenues par le générateur de benchmark.

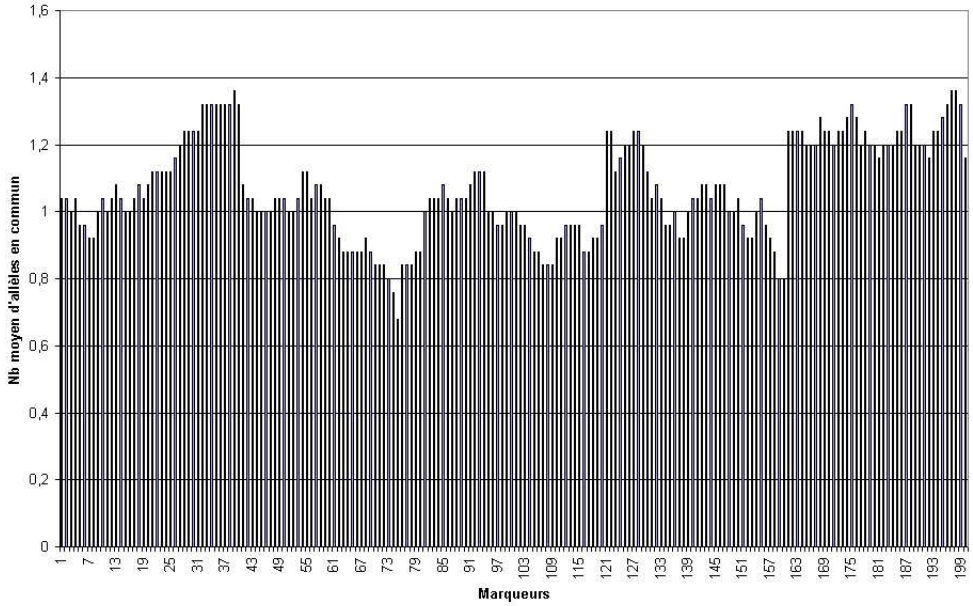


FIG. 3.8 – Exemple de partage allélique moyen sur les 20 paires d'individus générées par le benchmark.

liques pour l'exemple donné.

3.3 Étude du déséquilibre de liaison

Cette étude porte sur le déséquilibre de liaison (“linkage desiquilibrium”) pour les maladies multifactorielles et particulièrement sur l’étude d’associations de gènes candidats impliqués dans les maladies étudiées par le LGMM. L’objectif est de rechercher des configurations de variants d’ADN (marqueurs) fréquents chez les malades et non fréquents chez les témoins.

3.3.1 Présentation du problème

Nous allons tout d’abord définir quelques notions biologiques nécessaires à la compréhension du problème [BRLE98, Eti01].

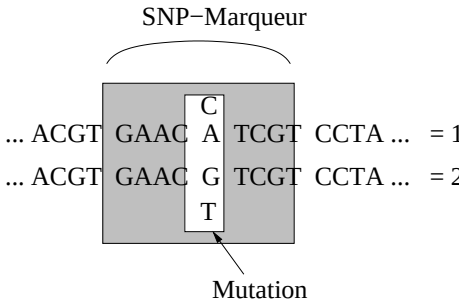


FIG. 3.9 – Schéma d’un marqueur et forme de SNP.

Une des voies de recherche de la génomique est l’étude des variations affectant la séquence d’ADN (polymorphisme génétique). Ainsi, les SNPs (Single Nucleotides Polymorphism), polymorphismes de nucléotides (voir figure 3.9) touchant un seul nucléotide (mutations ponctuelles isolées), sont des régions du génome variables d’une personne à l’autre. Chaque être humain posséderait 10 millions de SNPs, qui sont autant de variations ponctuelles et dont la combinaison fait de chaque individu une personne unique. Les SNPs localisés entre les gènes n’ont aucune incidence apparente sur l’homme. En revanche, ils peuvent être directement responsables de troubles fonctionnels s’ils sont situés dans les gènes ou les régions de régulation des gènes. Le repérage de SNPs peut ainsi permettre la découverte de gènes de prédisposition à de nombreuses maladies telles que le diabète, l’hypertension artérielle ou la polyarthrite rhumatoide. Ils peuvent être mesurés par spectrométrie de masse ou par fluorescence.

Un *haplotype* est un ensemble constitué de plusieurs SNPs. Par exemple, sur la figure 3.10, nous avons représenté un haplotype constitué de 4 SNPs répartis sur la carte génétique. Cet haplotype a pour valeur 1221 versus 1122. Si le marqueur est un bon indicateur pour suivre un gène, un haplotype peut permettre de suivre plusieurs gènes simultanément, c’est ce qui nous intéresse ici car dans le cas de l’étude des maladies multifactorielles, la survenue ou non d’une maladie dépend de plusieurs gènes. Il peut aussi permettre de mieux caractériser un SNP fonctionnel inconnu

Équilibre de liaison [Ali] : deux loci sont en équilibre de liaison si dans l’ensemble des gamètes de la population considérée et pour tout SNP A et pour tout SNP B situé entre ces deux loci, la fréquence de l’haplotype AB est égale au produit des fréquences des SNPs A et B. Au contraire,

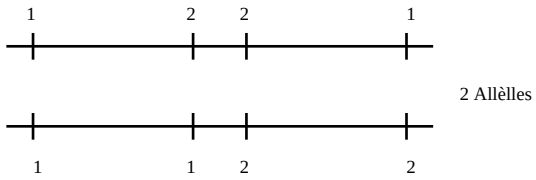


FIG. 3.10 – Exemple d'haplotype.

on parle de déséquilibre de liaison s'il existe des associations préférentielles entre SNPs.

Concrètement, si une maladie se déclenche uniquement en présence de deux variants particuliers, alors on observera un déséquilibre de liaison entre certains marqueurs chez les malades, déséquilibre que l'on ne retrouvera pas chez les personnes non-atteintes.

Nous illustrons le calcul du déséquilibre de liaison pour la HLA (Human Leucocyte Antigen) également nommée "complexe majeur d'histocompatibilité" (c'est-à-dire de compatibilité des tissus) ou CMH. Le HLA génère quatre marqueurs (A,B,C et D), dont chacun existe en plusieurs "versions". Considérons les fréquences alléliques $F(HLA-A) = 0.161$ et $F(HLA-C) = 0.153$. La fréquence probable de HLA-A / HLA-C sans déséquilibre de liaison est $0.161 \times 0.153 = 0.0246$. La fréquence observée pour HLA-A / HLA-C est de 0.089. Le déséquilibre de liaison est alors de $D = 0.089 - 0.0246$ soit 0.064. Le déséquilibre de liaison est alors testé avec un χ^2 pour évaluer la significativité de la différence de fréquences. Il existe trois variants de SNP codés 11, 21 (ou 12) et 22.

3.3.2 Données

Nous avons appliqué notre méthode sur des données provenant du Laboratoire de Génétique des Maladies Multifactorielles de l'Institut de Biologie de Lille. Les personnes pour lesquelles nous disposons de données appartiennent à deux groupes :

- les individus atteints : ces individus sont sélectionnés car la maladie a été diagnostiquée chez eux ;
- les individus témoins : ces individus sont non atteints par la maladie et ils sont utilisés pour tester les associations entre les personnes affectées et les personnes non affectées.

Le tableau 3.2 montre un exemple de données disponibles pour les individus. L'étude rapportée ici est composée de 132 individus pour 51 SNPs.

Individu \ SNP	**SNP**$_1$	...	**SNP**$_n$	**STATUT**
IND$_1$	11	...	22	Non atteint
IND$_2$	11	...	12	Atteint
...	...	...	...	...
IND$_{k-1}$	12	...	?	Non atteint
IND$_k$	11	...	22	Inconnu

TAB. 3.2 – Données sur les individus.

SNP	Fréq. de 1	Fréq. de 2
1	0.997	0.003
2	0.856	0.144
...	...	...
n-1	0.576	0.424
n	0.389	0.611

TAB. 3.3 – Fréquence des SNPs.

SNP_1	SNP_2	Déséquilibre
1	**2**	-1.00
1	**3**	0.67
...	...	...
n-1	**n**	0.42

TAB. 3.4 – Déséquilibre entre SNPs.

Deux autres tableaux nous donnent des informations sur les SNPs. Le tableau 3.3 indique pour chaque SNP la fréquence de chacun de ses variants (1 et 2). Ce tableau peut être directement calculé à partir du tableau 3.2, ou à partir d'une étude faite sur un plus grand nombre de personnes pour éviter d'être trop dépendant de l'échantillon de personnes. Le tableau 3.4 donne le déséquilibre entre chaque couple de SNPs. Ces deux tableaux sont nécessaires pour vérifier si un haplotype respecte les deux conditions sur la fréquence et le déséquilibre entre les SNPs (voir paragraphe 6.7.1).

3.3.3 Objectifs

L'objectif de cette problématique est de trouver des ensembles de SNPs (haplotypes) qui peuvent expliquer la maladie. Ces haplotypes peuvent être de différentes tailles en fonction du nombre de SNPs qui les composent.
Dans les études de déséquilibre de liaison, deux SNPs d'un haplotype doivent vérifier les deux conditions suivantes :

- leur déséquilibre deux à deux doit être inférieur à un seuil S_1 (voir tableau 3.4).
- la différence entre les plus petites fréquences de leurs deux variants (voir tableau 3.3) doit être plus grande qu'un seuil S_2 .

En général, il n'y a pas qu'un seul haplotype vérifiant toutes ces contraintes et permettant de séparer les personnes atteintes avec certitude.

Dans un premier temps, les biologistes ont décidé d'utiliser les logiciels procéduraux EH-DIALL et CLUMP pour évaluer un haplotype. Ces deux procédures sont basées sur des calculs statistiques et ils sont souvent utilisés dans la littérature pour évaluer les haplotypes. Dans cette première étude, l'objectif est donc de trouver des haplotypes répondant aux critères des fonctions EH-DIALL et CLUMP que nous présenterons dans les paragraphes suivants. Nous devons donc analyser le comportement de ces fonctions pour pouvoir mettre en place une méthodologie d'exploration des haplotypes.

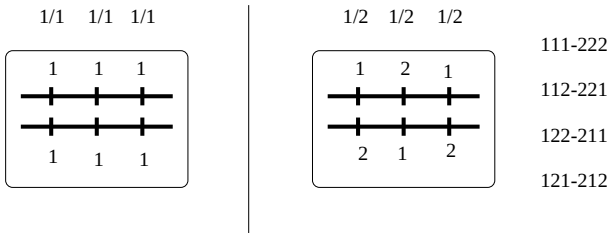


FIG. 3.11 – EH-DIALL : un exemple.

3.3.3.1 EH-DIALL

EH-DIALL [TO94] (EH pour Estimated Haplotype) est un programme qui détermine la distribution la plus probable des allèles dans un haplotype en fonction des valeurs de SNPs.
Soit un échantillon contenant un grand nombre d'individus pris aléatoirement dans la population, EH-DIALL estime les fréquences alléliques pour chaque marqueur. Les fréquences haplotypiques sont estimées avec une association allélique (Hypothèse H_1) et sans (Hypothèse H_0).
Par exemple, sur la gauche de la figure 3.11, un haplotype constitué de 3 SNPs de valeurs 1/1, 1/1 et 1/1, admet une unique distribution entre les 2 chromosomes ; c'est l'haplotype 111 versus 111. Sur la droite de la figure, l'haplotype avec 3 SNPs (1/2, 1/2 et 1/2) admet 4 distributions entre les chromosomes (111-222, 112-221, 122-211, 121-212). La tâche de EH est de choisir la distribution la plus probable entre toutes, ici 121 versus 212.
Ce programme est lancé séparément sur les individus atteints et non atteints.

3.3.3.2 CLUMP

CLUMP [SC95] est un programme permettant d'évaluer la signification des valeurs observées dans une table de contingence aux valeurs conditionnelles prévues sur les totaux marginaux. L'implémentation utilisée travaille avec des tables de taille 2 $\times$ N et a été modélisée pour l'utilisation dans l'étude d'association à partir de cas de contrôle en génétique. La signification est estimée en utilisant une approche de type Monte-Carlo, en effectuant des simulations répétées pour générer des tables ayant les même totaux marginaux que celle prise en considération et en comptant le nombre de fois où la valeur du chi-deux associée à la table de données réelles est atteinte par les données simulées aléatoirement. Ces niveaux de signification doivent être non biaisés (il doit y avoir suffisament de génération de tables).
CLUMP produit différentes statistiques : T_1 (qui est le résultat d'un test de χ^2), T_2, T_3 et T_4.
Chacune de ces mesures est associée à une p-value qui donne le niveau de confiance du résultat [4].
D'après les auteurs de [SC95], T_2 et T_3 ne sont pas pertinentes pour les problèmes considérés. Pour choisir la meilleure mesure parmi T_1 et T_4 , nous avons généré aléatoirement des milliers de tables de contingence simulant un dénombrement d'individus malades et sains possédant un haplotype donné pour un ensemble de SNPs.
Pour chaque table, les valeurs T_1 et T_4 sont calculées et représentées sur les figures 3.12 et 3.13. La figure 3.12 (resp.3.13) classe les tables par ordre croissant de T_1 (resp. T_4).

[4] Ceci est dû au fait que le logiciel utilise une méthode de Monte-Carlo pour diminuer le volume des calculs, le résultat n'est donc pas juste mais il a une certaine confiance.

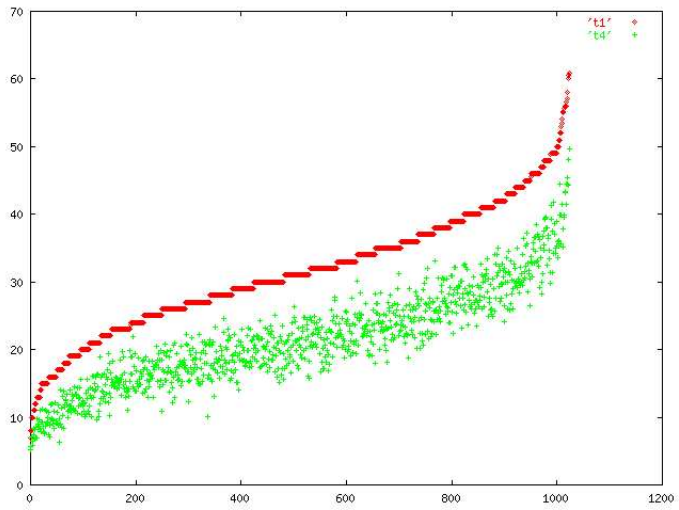


FIG. 3.12 – Comparaison entre T_1 et T_4, classé selon T_1 croissant.

Comme le montre ces figures, les deux méthodes donnent des résultats convergents, maximiser T_1 revient à maximiser T_4, nous avons donc choisi d'utiliser uniquement T_1, qui est plus rapide à calculer.
Ensuite, la deuxième contrainte est de minimiser la p-value, pour avoir confirmation de la qualité de la solution. Nous avons donc également généré des instances pour voir si on peut trouver une relation entre T_1 et sa p-value P_1. Les résultats sont exposés sur la figure 3.14 (graphique représentant pour chaque instance la valeur de T_1 et de 10*P_1, classé selon les valeurs de T_1 croissantes). On voit bien que lorsque l'on maximise T_1, la p-value diminue et atteint rapidement une valeur nulle, donc par la suite nous considérerons pour acquis cette propriété (ceci est un point très important car le calcul de la p-value coûte très cher, de l'ordre d'une minute par évaluation pour avoir une valeur précise).

La statistique qui correspond à notre problème est noté T_1. Un haplotype potentiellement intéressant pour cette étude sera hautement corrélé avec la maladie et aura donc une grande valeur de T_1.

3.3.3.3 Processus d'évaluation

Le figure 3.15 représente le processus d'évaluation pour un haplotype. A partir d'un ensemble de SNPs candidats, on estime dans un premier temps indépendemment, pour les personnes atteintes ou non, la distribution des allèles dans l'haplotype grâce à EH-DIALL. Puis, le programme CLUMP évalue l'association haplotype-maladie.

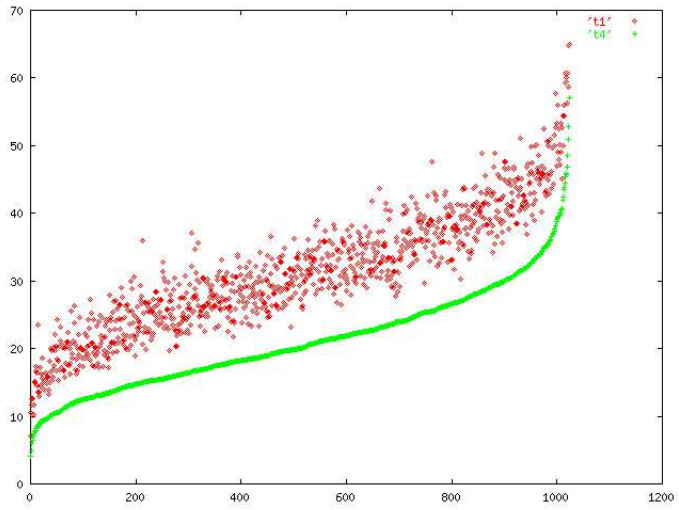


FIG. 3.13 – Comparaison entre T_1 et T_4, classé selon T_4 croissant.

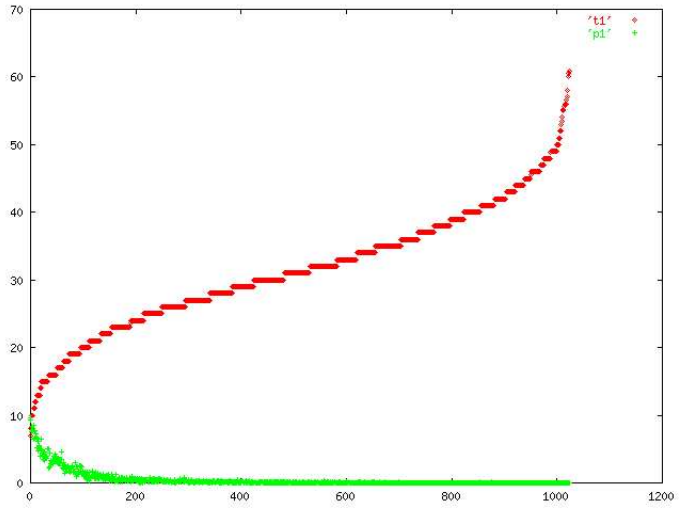


FIG. 3.14 – Comparaison entre T_1 et sa p-value P_1.

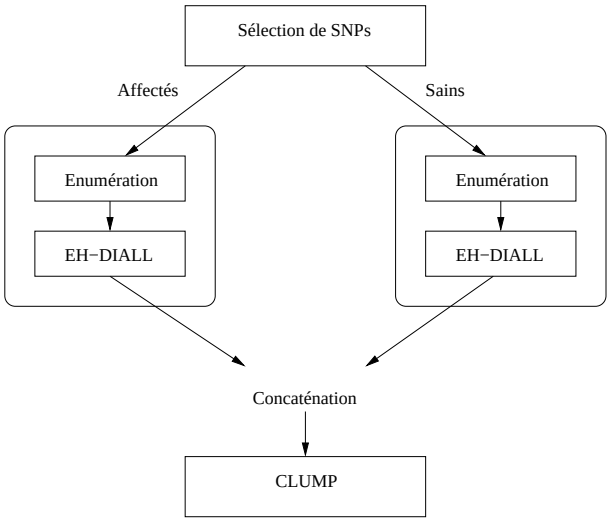


FIG. 3.15 – Processus d'évaluation d'un haplotype.

Dans un second temps, nous essaierons de traiter ce problème comme un problème d'extraction de connaissances où l'objectif est d'extraire des associations de SNPs (donc des haplotypes) permettant d'expliquer la maladie ce qui peut correspondre à une recherche de règles d'associations. Nous rajouterons des contraintes biologiques pour simuler les contraintes sur le déséquilibre de liaison et sur la différence de fréquence rencontrées dans cette problématique.

3.3.4 État de l'art sur l'étude du déséquilibre de liaison

L'analyse d'haplotypes par déséquilibre de liaison est une voie intéressante pour explorer les facteurs génétiques des maladies complexes. Des données commencent à être disponibles et leurs études sont relativement récentes.
Plusieurs méthodes statistiques pour détecter des déséquilibres de liaison [Ter95, DRR96, Laz98] ont été menées ces dernières années. Ces études proposent des modèles statistiques du déséquilibre de liaison autour d'un gène de susceptibilité de la maladie. Ces méthodes peuvent seulement considérer des associations d'une région à la fois et faire un certain nombre de suppositions sur le modèle biologique de la maladie.
Des auteurs ont également proposé des méthodes recherchant des modèles récurrents pour trouver des haplotypes associés à la maladie [TOV$^+$00, TOO$^+$00].
Dans ces articles, les auteurs présentent une méthode pour tracer le déséquilibre de liaision : haplotype pattern mining (HPM). Cette méthode, inspirée par des méthodes d'extraction de données, est basée sur la découverte des modèles récurrents (pattern). Ils définissent une classe des modèles utiles d'haplotype dans des données génétiques de contrôle et recherchent les haplotypes associés à la maladie. Toivonen et al. emploient un modèle statistique non paramétrique et ne font "aucune hypothèse au sujet du modèle de transmission de la maladie ".
Dans [LST$^+$01], les auteurs présentent un modèle bayesien appelé BLADE [RR02] qui produit des distributions de probabilité pour une maladie en utilisant l'inférence bayésienne. Ils emploient également des méthodes de chaîne de Markov et de Monte-Carlo pour intégrer la généalogie de la

mutation inconnue.

3.4 Conclusion

Ces deux problématiques vont être au cœur de notre travail. Elles ont en effet servi de point de départ pour les approches par méthodes d'optimisation sur les travaux d'extraction de connaissances qui ont été réalisés. Bien que l'objectif ne soit pas de seulement résoudre ces problèmes, il faut garder à l'esprit que la majorité des approches que nous développerons dans cette thèse avait comme objectif initial de fournir aux biologistes des outils leur permettant d'analyser leurs données et donc d'expliquer par cette approche des maladies comme le diabète et l'obésité.

Chapitre 4

La sélection d'attributs par méthodes d'optimisation

Dans ce chapitre, nous présentons l'approche évolutionnaire réalisée pour traiter des problèmes de sélection d'attributs relatifs aux problématiques réelles présentées au chapitre 3. Nous décrivons dans un premier temps la tâche d'extraction de connaissances considérée et les algorithmes classiques de la littérature existants. Nous présentons ensuite une première approche pour résoudre le problème décrit au chapitre 3 relatif à l'étude des paires de germains ("Identical by Descent"). Dans le cadre de cette application, nous comparons des opérateurs naturels induits par le codage utilisé à des opérateurs dédiés et nous décrirons des mécanismes avancés permettant un meilleur parcours de l'espace de recherche notamment en introduisant le parallélisme comme mécanisme de diversification. Nous effectuerons ensuite différentes expérimentations sur différents jeux de données aussi bien artificiels que réels. Dans un troisième temps, nous présenterons le travail effectué pour traiter le déséquilibre de liaison qui prend en compte la fonction d'évaluation demandée par les biologistes. Nous montrerons les particularités de ce problème en terme de paysage et de structure induite par la fonction d'évaluation et nous montrerons les choix que nous avons dûs faire afin de trouver une méthode de résolution efficace. Nous détaillerons notamment les mécanismes adaptatifs que nous avons développés. Nous introduirons un autre type de parallélisme que celui utilisé dans l'étude des paires de germains. Les résultats que nous présenterons sont obtenus sur les jeux de données réels provenant du Laboratoire de Génétique des Maladies Multifactorielles de l'Institut de Biologie de Lille.

4.1 Synthèse sur la sélection d'attributs

La sélection d'attributs consiste, dans un problème d'extraction de connaissances où un attribut représente un élément descriptif d'un objet, à réduire l'ensemble des attributs considérés. Ceci peut augmenter la précision de la prédiction ou réduire le temps de traitement des données.
Classiquement, la sélection d'attributs est définie coomme le fait de sélectionner un sous-ensemble de M attributs à partir d'un ensemble N, tel que $M < N$ et que la fonction critère choisie soit optimale sur le sous-ensemble de taille M choisi.
L'étude de ce problème se justifie facilement par le fait qu'une recherche exacte a un coût exponen-

tiel en temps de calcul et en espace mémoire. En effet, cela fait partie des problèmes NP-difficiles [NF77] et la sélection d'un sous-ensemble d'attributs demanderait l'exploration de tout l'espace de recherche. Pour N attributs, la recherche exhaustive consiste à explorer $2^N - 1$ sous ensembles possibles (cf. Figure 4.1). La recherche d'un sous ensemble de i attributs parmi N consiste à appliquer le critère d'évaluation C_N^i fois soit $\frac{N!}{i!(N-i)!}$ fois. Si on trouve M ensembles, on aura donc une complexité de $\sum_{i=0}^{M} C_N^i = O(N^M)$. Pour remédier à cela, le recours à des heuristiques est donc nécessaire.
Les méthodes existantes pour la sélection d'attributs utilisent des connaissances dans divers domaines : les statistiques, l'apprentissage, les heuristiques et les métaheuristiques, ...

Les algorithmes de sélection d'attributs peuvent être classés en deux catégories selon que la sélection d'attributs est faite indépendamment ou non de l'algorithme d'apprentissage supervisé sur lequel est basé le classifieur (cf. Figure 4.2). Il existe dans la littérature de nombreux travaux

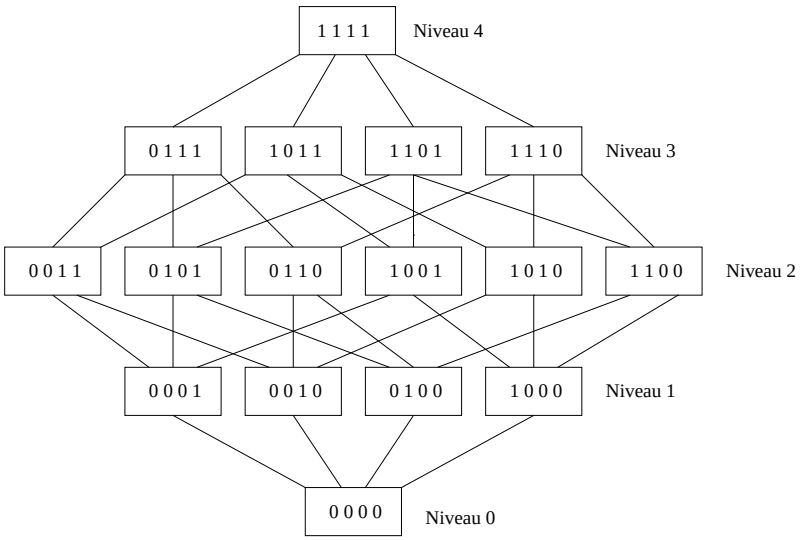


FIG. 4.1 – Treillis de l'espace de recherche.

sur la sélection d'attributs pour la classification et relativement moins pour le clustering. Pour permettre au lecteur une bonne compréhension des problèmes, nous présenterons les méthodes de sélection d'attributs pour la classification dans un premier temps, puis les méthodes de sélection d'attributs pour le clustering.

4.1.1 La sélection d'attributs en classification

L'objectif de cette tâche est de réaliser la sélection d'un sous-ensemble d'attributs pour améliorer la classification d'un ensemble d'instances définies par des attributs A et appartenant à une classe Y. Dans un premier temps, nous donnerons le vocabulaire relatif à la sélection d'attributs. Puis nous présenterons différents algorithmes classiques de la littérature et présenterons une brève

comparaison de leurs avantages et inconvénients.

4.1.1.1 Définitions pour la sélection d'attributs en classification

Nous allons tout d'abord donner un ensemble de définitions permettant de présenter les notions nécessaires à la compréhension de la tâche de sélection d'attributs.

Définition 7 *Sous-ensemble optimal (Kohavi et John [KJ98])*
Soit un inducteur I, et un ensemble de données d'entraînement D avec les attributs $A_1, A_2, ..., A_n$. Un sous-ensemble d'attributs optimal, A_{opt} est le sous-ensemble qui maximise la qualité du classifieur induit Cl=I(D), mesurée par le pourcentage de prédiction correcte sur un ensemble test.

Un sous-ensemble d'attributs optimal n'est pas nécessairement unique puisqu'il est possible d'avoir la même exactitude de classification en utilisant différents sous-ensembles de données.

Il existe plusieurs définitions de la pertinence d'un attribut dans la littérature. Celles-ci dépendent de la nature des données, de l'existence de bruit dans les données et de données dupliquées.
La définition suivante donnée par Almuallin et Diettrich en 1991 considère que tous les attributs sont des booléens [AD91].

Définition 8 *Un attribut A_i est dit pertinent pour un concept C si A_i apparaît dans chaque formule booléenne qui représente C. Il est dit non pertinent sinon.*

Définition 9 *Un attribut A_i est pertinent, s'il existe deux instances (X_1, Y_1) et (X_2, Y_2) appartenant à deux classes différentes ($y_1 \neq y_2$) telles que : la valeur de l'attribut A_i est différente pour les deux instances et les valeurs des autres attributs sont identiques.*

Gennari, Langley et Fisher (1989) considèrent des données bruitées et pouvant être dupliquées. Ils définissent les attributs pertinents comme ceux dont les valeurs changent systématiquement en fonction de l'appartenance de la donnée à telle ou telle catégorie. On peut l'exprimer formellement comme suit :

Définition 10 *Un attribut A_i est pertinent si et seulement si il existe une valeur a_i de cet attribut et une valeur de la classe Y, y, vérifiant $p(A_i = a_i) > 0$ et tels que : $P(Y = y|A_i = a_i) \neq P(Y = y)$. Avec $p(A_i = a_i)$ qui exprime le fait que les données contiennent au moins une instance ayant la valeur a_i pour l'attribut A_i.*

Cette formule nous permet de dire que A_i est pertinent si la probabilité de prédire Y, en connaissant le valeur de A_i, est différente de la probabilité de prédire Y sans connaître la valeur de A_i.
Dash et Liu [DL98] définissent un attribut non-pertinent comme celui n'affectant pas la structure fondamentale des données et un attribut redondant comme celui n'apportant rien de nouveau pour décrire la structure fondamentale des données.
Dans leurs travaux, Kohavi et John reformulent l'importance d'un attribut dans un contexte probabiliste, où la probabilité de distribution est supposée connue [KJ98]. Dans les définitions suivantes,

S_i représente l'ensemble de tous les attributs de description excepté l'attribut A_i et s_i représente l'ensemble de valeurs de S_i.

Définition 11 *Pertinence forte d'un attribut (Kohavi et John [KJ98])*
Un attribut A_i est fortement pertinent si et seulement si il existe a_i, y et s_i pour lesquels $p(A_i=a_i,S_i=s_i)>0$ tels que : $p(Y=y \mid A_i=a_i, S_i=s_i) \neq p(Y=y \mid S_i=s_i)$

Cette formule nous permet de dire que A_i est très pertinent si la probabilité de prédire Y en connaissant la valeur de A_i et les valeurs des autres attributs de S_i est différente de prédire Y en connaissant seulement des valeurs des autres attributs de S_i.

Le processus de sélection d'attributs comporte plusieurs étapes. Son espace de recherche peut être modélisé sous forme d'un treillis de gallois (cf. figure 4.1 pour un exemple à quatre attributs) où un 1 représente la sélection d'un attribut et un 0 la non sélection. Un niveau représente un ensemble d'états ayant le même nombre d'attributs sélectionnés.

Définition 12 *Une méthode de sélection de variables est composée généralement des composantes suivantes (Leray [Ler98]) :*

- *un critère d'évaluation pour comparer différents sous-ensembles de variables et en retenir un,*
- *une procédure de recherche pour explorer différentes combinaisons de variables,*
- *un critère d'arrêt pour stopper la procédure de recherche ou déterminer l'ensemble des variables à sélectionner.*

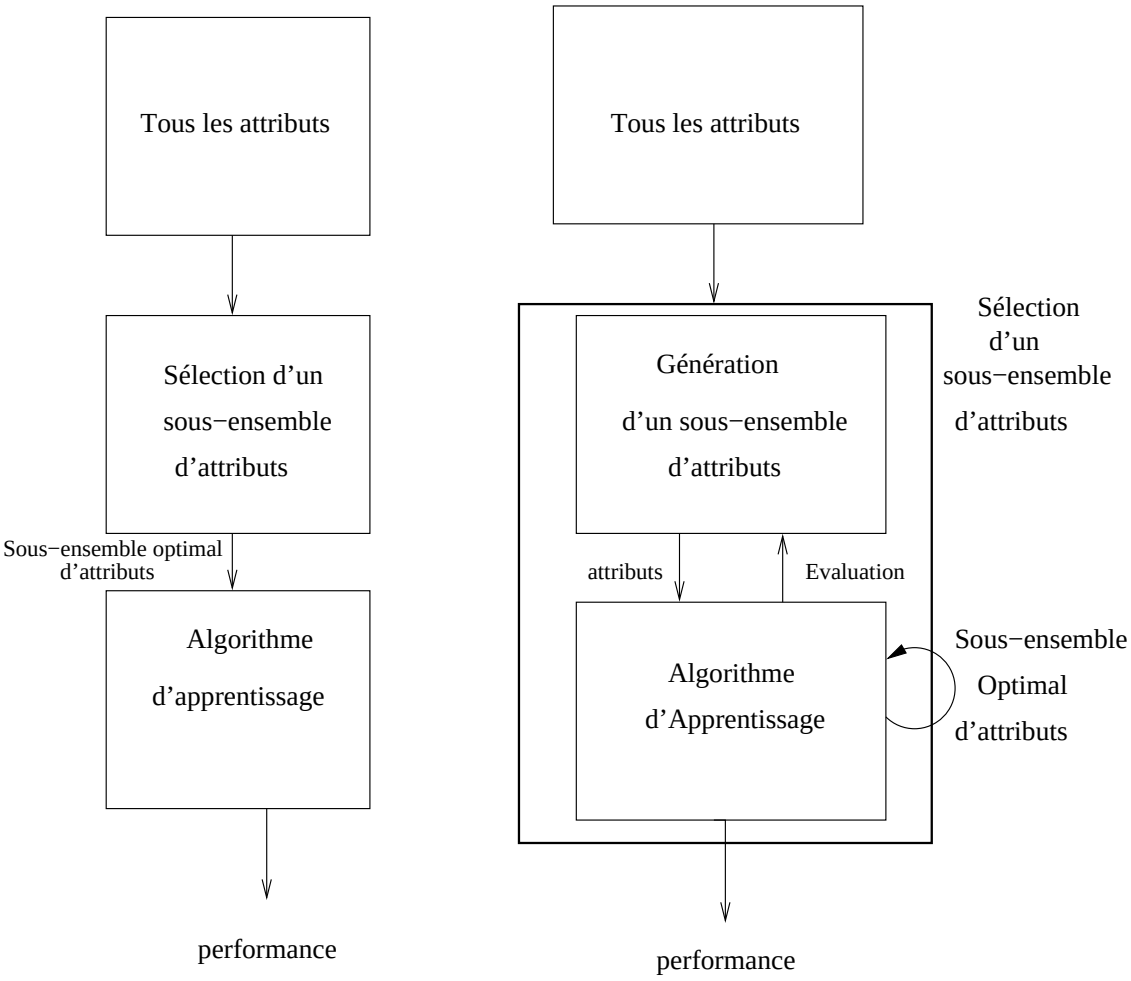
4.1.1.2 Méthodes pour la classification

Nous présentons les différentes méthodes selon la stratégie d'évaluation qu'elles utilisent. Nous verrons que la stratégie d'évaluation peut être de deux types : l'approche de type enveloppante (wrapper) qui utilisent le classifieur pour évaluer le sous-ensemble d'attributs sélectionnés ou de type filtrante (filter) qui utilise une fonction spécifique pour évaluer le sous-ensemble d'attributs sélectionnés (voir figure 4.2).

4.1.1.2.1 Méthodes filtrantes

Ces méthodes sélectionnent des variables en utilisant différentes approches et différents critères pour calculer la pertinence d'une variable avant le processus d'apprentissage, i.e. la construction du classifieur.
Nous détaillerons ici la phase de sélection de M attributs parmi N des méthodes présentées.

FOCUS
L'algorithme FOCUS introduit par Almuallim et Diettrich [AD91, AD94] est de type Stepwise Forward Selection.
Description : Exploration exhaustive de tous les sous-ensembles de variables afin de déterminer le plus petit sous-ensemble qui est suffisant pour déterminer l'appartenance à une classe de toutes les instances.
Désavantage : Originellement définie pour des données booléennes non bruitées, l'algorithme de



(a) Approche Filter (b) Approche Wrapper

FIG. 4.2 – Les deux approches de sélection d'attributs.

base est restreint à 2 classes. De plus Dash et Liu [DL98] indiquent que l'algorithme prend du temps si la taille du sous ensemble reste importante. FOCUS a en effet une complexité en temps de $O(N^M)$.

L'algorithme RELIEF
L'algorithme RELIEF a été présenté par Kira et Rendell [KR92b, KR92a].
Description : Cette méthode calcule pour chaque attribut sa pertinence en comparant une instance et sa classe correspondante sur plusieurs sous ensembles de données. RELIEF est un algorithme à processus aléatoire. En effet, il échantillonne les instances de façon aléatoire à partir de l'ensemble de données d'exemples et met à jour les valeurs de pertinences basées sur les différences entre l'instance sélectionnée et les deux plus proches instances de la même classe et d'une classe opposée ("near-hit" et "near-miss"). A l'origine, il a été réalisé pour l'algorithme d'induction K plus proches voisins.
Désavantages : RELIEF ne tient pas compte d'une éventuelle redondance entre les variables ou d'une forte corrélation. Cette méthode dépend fortement du nombre d'exemples par classe.
Avantages : RELIEF est tolérant au bruit dans les données et n'est pas affecté par les interactions entre les attributs.

B&B
Tout d'abord développé par Fukunaga et Narendra [NF77], la méthode de séparation et d'évaluation ("Branch and Bound") pour la sélection d'attributs a été repris par Lui, Dash et Motoda [DL98]. Les fonctions d'évaluation généralement utilisées sont : la distance de Mahalanobis, la fonction discriminante, le critère de Fisher, la distance de Bhattacharya et la divergence (par exemple celle de Kullback-Liebler).
Description : L'algorithme commence avec l'ensemble total d'attributs et enlève un attribut à la fois. L'algorithme garantit d'arriver à la solution optimale sans parcourir tous les sous-ensembles de variables.
Désavantages : BB est inefficace pour réduire l'espace de recherche. La fonction d'évaluation doit être monotone.

LVF
LVF [DL98, LS96] est un algorithme probabiliste, dit "Las Vegas algorithm", pour la sélection d'attributs.
Description : LVF fait des choix probabilistes de sous-ensembles dans la recherche de l'ensemble optimal. LVF garde le plus petit sous-ensemble d'attributs généré aléatoirement dont le taux d'inconsistance satisfait un seuil. Il est rapide pour faire décroître le nombre d'attributs.
Il existe différentes formes de l'algorithme [CL99b, LS96].
Désavantage : LVF est aveugle et génère des sous-ensembles inintéressants. A cause d'un problème de dégradation [CL99b], LVF est de plus en plus lent au fur et à mesure qu'il s'approche d'une solution optimale.
Avantage : LVF trouve un sous-ensemble d'attributs même si il y a du bruit dans les données. De plus, l'utilisateur obtient rapidement un bon sous-ensemble.
Une version adaptative appelée WLVF a été proposée par Chen et Liu [CL99b].

Méthode	Habilité à prendre en compte/à produire						
	Type de données			Classes	Grande	Bruit	Sous-ensemble
	Continue	Discrète	Nominale	Multiples	BD		Optimal
FOCUS	N	O	O	O	N	N	O
Relief	O	O	O	N	O	O	N
LVF	N	O	O	O	O	O	O[1]
B&B	O	O	N	O	-	-	O[1]

TAB. 4.1 – Récapitulatif des principales méthodes filtrantes de sélection d'attributs présentées dans la littérature.

QBB
QBB est une version hybride entre ABB (cf 4.1.1.2.2) et LVF proposée par Dash et Liu [DL98]. Les auteurs utilisent tout d'abord LVF (recherche de type probabiliste) pour réduire le nombre d'attributs puis utilisent ABB. Ils limitent le temps d'utilisation de leur algorithme (ils ont donc un algorithme approximatif) en donnant à chaque phase la même durée d'exécution.

Le tableau 4.1 récapitule les principales méthodes de sélection d'attributs présentées dans la littérature.

4.1.1.2.2 Méthodes enveloppantes

Ces méthodes se servent de l'algorithme d'induction comme d'une boîte noire : l'apprentissage est effectué avec les variables sélectionnées et les performances sont estimées à partir de l'erreur de généralisation.
La méthode enveloppante conduit une recherche dans l'espace des paramètres possibles. Une recherche requiert [KJ98] :

- Un espace d'états où chaque état représente un sous ensemble d'attributs. Pour n attributs, il y a n bits dans chaque état et chaque bit indique si l'attribut est présent ou absent.
- Un état initial : lorsque l'on fait une sélection "forward", la recherche commence avec un ensemble vide d'attributs ; lorsque l'on fait une élimination "backward", la recherche commence avec l'ensemble complet d'attributs.
- Une condition d'arrêt.
- Une méthode de recherche.

SFS (Sequential Forward Selection)
SFS commence sans attribut puis évalue tous les sous-ensembles d'attributs avec exactement un attribut. SFS sélectionne celui avec la meilleur performance et ajoute à ce sous-ensemble, le sous-ensemble d'une taille plus grande.

SBS (Sequential Backward Selection)
SBS commence avec tous les attributs et enlève un attribut à la fois (celui qui, étant enlevé, provoque une amélioration maximale du critère).
Avantage : Très rapide
Désavantage : SBS est aveugle et n'arrive pas à travailler avec une fonction critère qui contient

[1] Sous certaines conditions.

des imprécisions ou des bruits.

Hill-Climbing
Nous reprenons ici le Hill-Climbing décrit par Kohavi et John [KJ97]. Ils utilisent comme fonction d'évaluation cinq validations croisées répétées de multiples fois. Le nombre est déterminé en regardant la déviation standard de l'exactitude estimée. Si la déviation standard de l'exactitude estimée est de 1% et si les cinq validations croisées n'ont pas toutes été exécutées, alors on exécute une autre validation croisée (cross-validation).

Best-first search
La méthode de recherche Best-first (meilleur d'abord) décrite par Kohavi et John [KJ97] est une méthode plus robuste que le Hill-Climbing. L'idée est de sélectionner le nœud le plus prometteur non encore développé.
Best-first search donne en général de meilleurs résultats que le Hill-Climbing.

ABB : Automatic Branch & Bound
L'algorithme ABB (Liu, Motoda et Dash [DL98]) palie à un des désavantages du B&B en permettant des fonctions d'évaluation qui ne sont pas monotones. La borne est le taux d'inconsistance des données. Le test de légitimité détermine si un sous-ensemble est un enfant d'une branche coupée en appliquant la distance de Hamming. La procédure "InConCal" calcule le taux d'inconsistance en s'assurant qu'il n'y a pas de sous-ensemble dupliqué généré et pas d'enfant d'une branche coupée.
Avantage : ABB enlève les attributs redondants, corrélés et non significatifs. Il est simple à implémenter et garantit des sous-ensembles optimaux d'attributs.
Désavantage : ABB est incapable de réduire l'espace de recherche.

Les auteurs Lui et Dash [DL97] proposent une hiérarchie des méthodes que nous avons utilisé pour classifier les méthode de sélection d'attributs présentée en annexe par la figure C.

4.1.1.2.3 Méthodes et résultats

Afin de pouvoir résoudre efficacement un problème, il est important de savoir quelle méthode de sélection de variables utiliser et dans quelles conditions.
Un point crucial dans ce choix réside dans l'utilisation que l'on veut en faire. Dans le contexte de l'extraction de connaissances, dans lequel nous positionnons notre travail, la taille des données en terme de nombre de variables et de nombre d'instances est de plus en plus volumineuse et il va de soit qu'un classifieur ne peut pas être utilisé directement. Les méthodes enveloppantes (wrapper) sont donc difficilement applicables. Ensuite, il faut se demander quels sont les objectifs de notre approche : augmenter la précision du classifieur, diminuer la complexité des représentations pour pouvoir extraire des règles plus facilement compréhensibles, diminuer le nombre de variables pour accélérer les processus d'extraction de connaissances mis en jeu. Le tableau donné en annexe C.1 résume les résultats de la littérature des différentes méthodes que nous avons exposé dans cette partie.

4.1.2 La sélection d'attributs pour le clustering

Le clustering ou catégorisation est une tâche importante en extraction de connaissances qui rassemble des groupes d'objets similaires. Comme le clustering est réalisé sur des données dont on ne connaît aucune information sur leur classe d'appartenance, les algorithmes traditionnels de sélection d'attributs pour la classification présentés précédemment ne peuvent pas être utilisés.

4.1.2.1 Importance des attributs dans le clustering

La plupart des méthodes de clustering font l'hypothèse que tous les attributs ont la même importance pour le clustering et ils ne font aucune différence entre les différents attributs. C'est une des raisons pour lesquelles la plupart des méthodes de clustering ne sont pas performantes dans le cas des données de grandes tailles.
Dans la réalité, différents attributs ont des effets différents sur le clustering. Un attribut important aide dans la création de clusters alors qu'un attribut non important peut être assimilé à du bruit et peut être enlevé pour réduire la taille des données et ainsi rendre plus efficace le clustering. Alors que la sélection d'attributs pour la classification supervisée a été largement étudiée, relativement peu de travaux existent pour la sélection d'attributs pour le clustering. A notre avis, il existe deux problèmes majeurs pour transformer une méthode pour la sélection d'attributs pour la classification en une méthode pour le clustering. Tout d'abord, l'absence de label de classe rend impossible l'utilisation des mêmes fonctions d'évaluation que dans le cas de la sélection d'attributs pour la classification. De plus, il n'existe pas de critère standard pour évaluer un sous-ensemble d'attributs sélectionnés.
Certains auteurs utilisent le label de classe a posteriori pour mesurer la performance de méthodes de clustering [Tal00, DLY97, DR97]. D'autres auteurs utilisent des indicateurs de performance dont nous allons présenter les plus connus.
Dans [DL00], Dash et Liu proposent une méthode de sélection d'attributs pour le clustering basée sur l'entropie adaptée au clustering et sur une méthode de rang. Le calcul de l'entropie demande une connaissance des classes auxquelles appartiennent les exemples. Les auteurs proposent de calculer l'entropie sans recourir à ce besoin de classe.
Soit X_i la $i^{\text{ème}}$ donnée, X_{ik} est la valeur du $k^{\text{ème}}$ attribut de la $i^{\text{ème}}$ donnée, F_k est le $k^{\text{ème}}$ attribut où i=1..N et k=1..N.
D_{i_1,i_2} et S_{i_1,i_2} sont respectivement la distance et la similarité entre X_{i_1} et X_{i_2}.
χ_j représente le $j^{\text{ème}}$ cluster où j=1..c.
La similarité, S_{i_1,i_2}, entre deux instances X_{i_1} et X_{i_2} est grande si les deux instances sont très proches l'une de l'autre et elle est faible si les deux instances sont éloignées (l'une de l'autre).
La similarité est basée sur une notion de distance (les auteurs utilisent la distance euclidienne pour les données de type numérique et la distance de Hamming pour les données de type nominal). Pour les données numériques, on peut définir mathématiquement la similarité comme étant $S_{i_1,i_2} = e^{-\alpha \times D_{i_1,i2}}$ où α est un paramètre.
La similarité sera donc forte pour des instances proches mais faible pour des points éloignés. Elle est discrétisée pour pouvoir traiter les variables de type numérique et de type nominal.
L'entropie entre deux points X_{i_1} et X_{i_2} est alors définie comme étant :
$E = -S_{i_1,i_2} \log S_{i_1,i_2} - (1 - S_{i_1,i_2}) \log(1 - S_{i_1,i_2})$ qui peut prendre comme valeur maximum 1.0 pour $S_{i_1,i_2} = 0.5$ et pour valeur minimale 0.0 pour $S_{i_1,i_2} = 0.0$ et $S_{i_1,i_2} = 1.0$.

Pour un ensemble de N instances, l'entropie est la suivante :

$$E = -\sum_{i_1=1}^{N}\sum_{i_2=1}^{N}(S_{i_1,i_2} \times \log S_{i_1,i_2} + (1 - S_{i_1,i_2}) \times \log(1 - S_{i_1,i_2})).$$

Dans leur travail, Liu et Dash fixe $\alpha = \frac{-\ln 0.5}{\overline{D}}$, où $\overline{D}$ qui représente la distance moyenne.
Ils utilisent ensuite un classement de tous les attributs en notant chaque attribut pour l'entropie calculée sur tous les attributs privés de celui considéré.

4.1.2.2 Méthodes pour le clustering

Nous présentons dans cette section les méthodes les plus connues de sélection d'attributs pour le clustering.
Dans [DR97], Devaney et Ram proposent une comparaison entre COBWEB et leur algorithme AICC.
COBWEB est un système de clustering qui représente des concepts probabilistes dans une structure d'arbre hiérarchique. Chaque ensemble de même parent dans la hiérarchie est appelé *partition* et la métrique utilisée, nommée "utilité de catégorie", mesure la capacité à prédire la valeur des attributs dans une partition par rapport à toute la base.
COBWEB construit ses concepts de façon incrémentale hiérarchique grâce à quatre opérateurs :

- *incorporate* pour ajouter une instance à un concept existant,
- *create* pour construire un nouveau concept contenant l'instance de la partition,
- *merge* qui essaie de combiner deux noeuds qui sont identifiés comme les meilleurs hôtes pour une nouvelle instance d'entraînement,
- *split* qui réalise l'opération inverse, c'est à dire qui tente de remplacer l'hôte par ses enfants pour spécialiser un concept.

COBWEB est alors utilisé dans le processus de sélection d'attributs comme évaluation pour une méthode forward ou backward en mesurant l' "utilité de catégorie" générée par COBWEB. La terminaison intervient quand le score n'est plus amélioré.

AICC (Attribut-Incremental Concept Creator) prend en entrée un concept hiérarchique et un nouvel attribut à ajouter ou à enlever. L'AICC réalise un HillClimbing dans l'espace des concepts comme COBWEB mais lorsqu'il y a un changement dans la base de données, AICC commence au point précédent de l'espace de recherche alors que COBWEB recommence entièrement la recherche. AICC permet alors d'arriver à un nouvel ensemble de façon plus rapide.
Dans le tableau C.2 donné en annexe, nous indiquons la performance obtenue pour différents jeux de données. Nous donnons les résultats de COBWEB et AICC avec les différents types de sélection possibles, forward (FS) ou backward (BS).

Dans un premier temps, nous avons présenté la sélection d'attributs pour la classification. Nous avons tout d'abord introduit les principales notions nécessaires à la compréhension du problème, puis nous avons présenté les méthodes classiques de sélection d'attributs pour la classification.
Dans un second temps, nous avons présenté la sélection d'attributs pour le clustering qui est une tâche relativement peu étudiée dans la littérature.

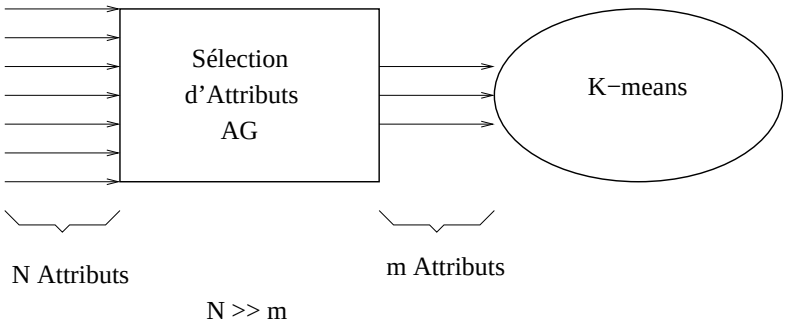


FIG. 4.3 – Notre approche en deux phases.

Après cette introduction à la sélection d'attributs, nous allons présenter notre approche par méta-heuristique que nous avons développée pour les problématiques introduites au chapitre 3.

4.2 Cas de l'étude des paires de germain ("Identical By Descent")

Dans le cadre de cette étude réalisée conjointement avec l'Institut de Biologie de Lille, nous désirons connaître les facteurs génétiques ou environnementaux que nous appellerons par la suite "attributs" impliqués dans des maladies de type multifactorielles présentées au chapitre 3.2. L'hypothèse fondamentale de notre recherche est qu'il n'y a pas qu'une seule combinaison d'attributs pouvant provoquer la maladie, mais un ensemble de combinaisons d'attributs dont certains peuvent appartenir à plusieurs combinaisons. Notre objectif est donc de fournir différents sous-ensembles d'attributs.
Ce travail a fait l'objet d'un chapitre du livre "Evolutionary computation in bioInformatics" [JDT02b], ainsi que d'un article dans un numéro spécial de la revue Knowledge Based Systems [JDTG02] et de plusieurs présentations dans des conférences [JDT01b] ou des workshop [JDT01a]. Une autre approche du problème, à l'aide d'un branch-and-bound, a été présentée dans la conférence EMGM [JDT+01c].
Notre approche va être divisée en deux phases (voir figure 4.3). Dans un premier temps, nous allons sélectionner les attributs pertinents grâce à un algorithme génétique. Nous allons ensuite présenter la seconde phase qui utilise le clustering comme simple validation.

4.2.1 Présentation de l'algorithme génétique

Nous présenterons dans les paragraphes suivants les principaux éléments de l'algorithme génétique proposé en insistant sur les apports des différents mécanismes.

4.2.1.1 Codage

Deux types de codage sont possibles pour la sélection d'attributs. Le premier consiste à se fixer un nombre maximum d'attributs à sélectionner v, et un individu représente une combinaison

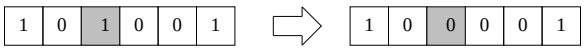


FIG. 4.4 – Exemple de mutation.

possible de v attributs. La seconde (celle que nous avons choisie) code un individu par une chaîne de n bits (n étant le nombre total d'attributs possibles). Le $i^{\text{ème}}$ bit peut valoir, soit 0, soit 1, et indique si le $i^{\text{ème}}$ attribut est sélectionné (valeur du bit à 1) ou s'il n'est pas sélectionné (valeur du bit à 0).
Exemple d'individu :

1	0	1	0	0	1

Cet individu nous indique que les attributs 1, 3, 6 sont sélectionnés. Il est a noté que dans notre cas, l'individu a une taille de 400 à 1200 bits suivant la taille du problème traité.

4.2.1.2 Opérateurs génétiques

Le codage simple binaire utilisé permet l'emploi d'opérateurs classiques de la littérature [Gol89] fonctionnant sur des chaînes de bits de longueur fixe.

Croisement : l'opérateur de croisement implémenté est l'opérateur de croisement deux points pour lequel les deux points de cassure sont générés aléatoirement.

Mutation : La mutation naturelle change un bit de façon aléatoire (voir figure 4.4). Un individu a une probabilité de mutation très élevée.

Les opérateurs classiques ou naturels sont les plus utilisés dans la littérature sur la sélection d'attributs par méthode d'optimisation mais ce n'est pas parce qu'on peut les utiliser qu'ils constituent la meilleure approche pour ce type de problème. Ainsi, depuis Radcliff [Rad91], certains d'auteurs travaillent sur l'intérêt de différents opérateurs naturels par rapport à des opérateurs adaptés à la problématique [GSW99, GSW98, CL99b].
Dans cette optique, nous proposons un autre croisement dédié au problème et nous avons modifié l'opérateur de mutation.
Nous avons donc réalisé un type de croisement qui est destiné à la sélection d'attributs : le croisement *Subset Size-Oriented Common Feature Crossover Operator* (SSOCF) [EHM00] dont l'objectif est de :

- Conserver les blocs d'information utiles.
- Mieux explorer le front des solutions non-dominées.
- Produire des enfants gardant les mêmes distributions que les parents.

Pour cela, chaque enfant conserve les attributs communs de leurs parents. Les attributs non partagés sont hérités par les enfants du i^{ème} parent avec la probabilité $\frac{(n_i - n_c)}{n_u}$, où n_i est le nombre

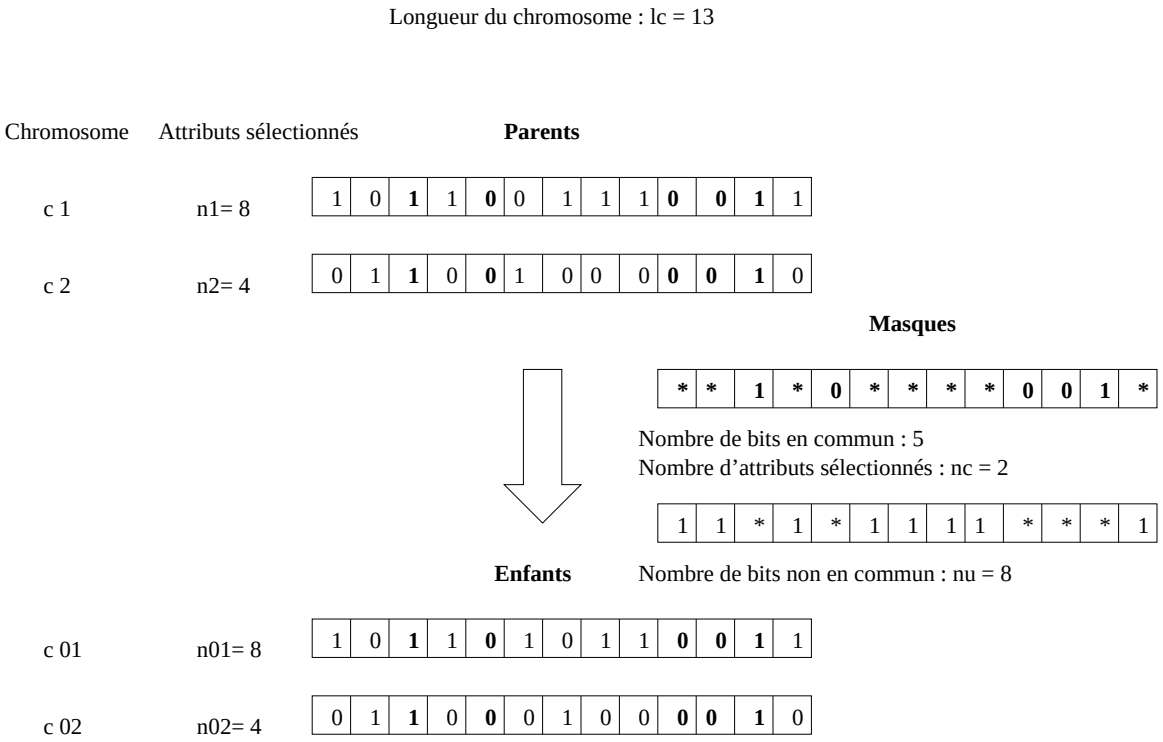


FIG. 4.5 – Exemple de reproduction par l'opérateur de croisement.

d'attributs sélectionnés du parent, n_c est le nombre d'attributs en commun entre les deux parents et n_u est le nombre d'attributs sélectionnés non partagés.
Comme la redistribution des bits correspondants à des attributs non-partagés se fait de façon aléatoire, le nombre d'attributs sélectionnés des enfants peut être différent de celui de leurs parents.

Durant la phase de mutation, un individu a une probabilité p_{mut} de muter. Pour pouvoir créer une grande diversité, nous utiliserons un fort taux de mutation.
Deux opérateurs de mutation sont choisis en fonction de l'amélioration des solutions (nombre d'individus mutés/gardés) et du nombre de cycles. Notre objectif est de tester la sélection de certains attributs sans trop augmenter le nombre d'attributs sélectionnés.
Le premier opérateur de mutation choisit un nombre n de bits à changer (n entre 1 et 5) : n bits sont alors aléatoirement choisis et changés selon une probabilité non symétrique, pour éviter la sélection d'un trop grand nombre d'attributs : pour changer un 1 en un 0 la probabilité est mise à 0.5, pour changer un 1 en un 0, la probabilité est mise à 1.
Le second opérateur choisit un bit à 1 et un bit à 0 : le 1 devient un 0, et le 0 devient un 1.
Nous comparerons dans la suite de ce chapitre les performances de ces différents opérateurs.

4.2.1.3 Fonction d'évaluation (fi tness)

Nous sommes ici dans un cas d'apprentissage non supervisé et nous ne disposons que d'observations non classées ou sans étiquette de classe. Comme nous l'avons observé précédemment, nous devons créer une fonction d'évaluation qui réalise ce que nous désirons : sélectionner les attributs pouvant être en corrélation et prendre en compte les spécificités biologiques.
Nous allons utiliser l'algorithme génétique pour sélectionner des groupes d'au moins deux attributs. La méthode du π_{moyen} nous donnera les attributs ayant la plus forte ressemblance pour une paire d'individus malades. Trouver une "bonne" fonction fitness est le point le plus important d'un algorithme génétique pour ce type de travail. La fonction que nous proposons est une aggrégation linéaire de deux critères. Le premier favorise un faible support et un faible nombre d'attributs sélectionnés (voir figure 4.8). Les biologistes pensent en effet que relativement peu d'attributs doivent être impliqués dans la maladie. La deuxième partie favorise un support élevé pour des ensembles d'attributs de grande taille (voir figure 4.7).
Cette fonction d'évaluation tend à favoriser les grands ensembles d'attributs lorsque le support est élevé.

$$F'_{évaluation} = \alpha \times \ln\left(2 + (1-S) \times \frac{\frac{Nbre\ total\ d'att}{10} - 10 \times Nbre\ d'att\ sélect.\ significatifs}{Nbre\ total\ d'att.}\right) + \beta \times \ln\left(2 + S \times \frac{\frac{Nbre\ total\ d'att.}{10} + 10 \times\ Nbre\ d'att\ sélect.\ significatifs}{Nbre\ total\ d'att}\right)$$

Où :

- Support : $S = \frac{|A \cap B \cap C ...|}{|A \cup B \cup C ...|}$
  - A, B, C ... représente les attributs sélectionnés,
  - $|\ A \cap B \cap C ...\ |$ représente le nombre de paires où les attributs sélectionnés sont simultanément à 1 dans la matrice Paire/IBD,
  - $|\ A \cup B \cup C ...\ |$ représente le nombre de paires où il y a au moins un des attributs présents,
- α et β sont des paramètres permettant de donner plus ou moins d'importance à un membre de la fonction d'évaluation,
- *Nombre d'attributs sélectionnés significatifs* représente le nombre d'attributs sélectionnés n'ayant pas de voisins sélectionnés à une distance inférieure ou égale à d. La distance d est fixée en fonction du problème et le voisinage s'arrête lorsqu'il y a un changement de chromosome. Soit la fonction de calcul :

$$Nbre\ d'attributs\ sélectionnés\ significatifs = \sum_i P\Big(\sum_{j=1}^{d} \delta_{i,j}\Big)$$

$$\text{avec } \delta_{i,j} = \begin{cases} 1 \text{ si } x_i = x_j \\ 0 \text{ sinon} \end{cases}$$

$$\text{et } P(x) = \begin{cases} 1 \text{ si } x = 0 \\ 0 \text{ si } x > 0 \end{cases}$$

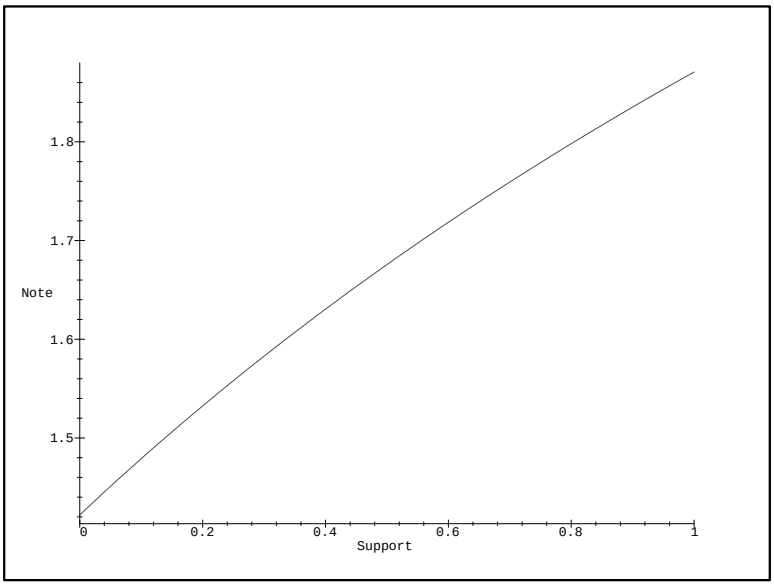


FIG. 4.6 – Evolution de la fonction d'évaluation en fonction du support.

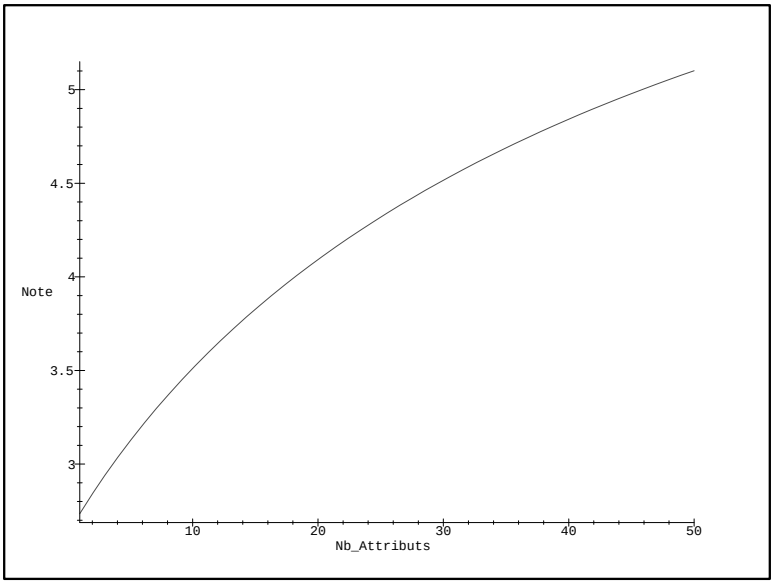


FIG. 4.7 – Evolution de la fonction d'évaluation en fonction de nombre d'attributs sélectionnés intéressants pour un support fort (0.9).

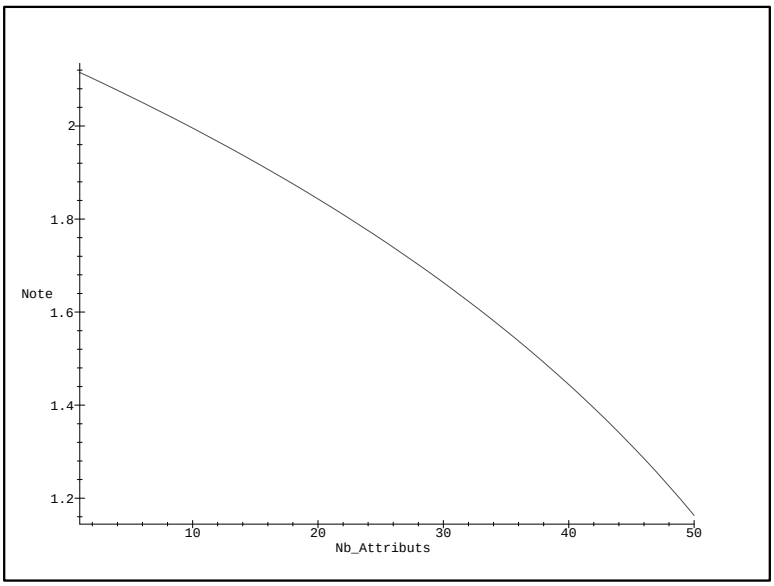


FIG. 4.8 – Evolution de la fonction d'évaluation en fonction de nombre d'attributs sélectionnés intéressants pour un support faible (0.2).

Exemple :
Soit l'individu suivant :

1	1	0	0	0	1	1	1	0

où | représente une coupure chromosomique.
Cet individu aura trois attributs significatifs pour d=1.

Les figures 4.6, 4.7, 4.8 montrent le comportement de la fonction d'évaluation dans différents cas. Notre fonction est corrélée positivement au support. Nous voulons qu'entre deux ensembles de support élevé, l'ensemble ayant le plus d'attributs soit favorisé ce qui est notre cas comme le montre la figure 4.7 alors que dans le cas d'un support faible nous favorisons les ensembles ayant un faible nombre d'attributs comme le montre la figure 4.8.
De plus, pour faciliter la comparaison entre les valeurs de la fonction d'évaluation des individus, nous effectuons une mise à l'échelle linéaire de la fonction d'évaluation :

$$F_{évaluation} = 10 \times F'_{évaluation}$$

Exemple :
Prenons par exemple le tableau suivant :

Paire \ IBD :	A	B	C	D	E
p_1	1	0	1	1	0
p_2	1	0	0	1	1
p_3	1	0	1	1	0
p_4	1	0	1	0	1
p_5	1	1	0	0	0

Fixons d=2 pour cet exemple.
Pour l'individu : | 1 | 1 | 0 | 0 | 0 |
$S = \frac{1}{5}$, Nbre d'attributs sélectionnés significatifs $= 1$

$$F'_{évaluation} = ln\Big(2 + (1 - \frac{1}{5}) \times \frac{\frac{5}{10} - 1 \times 10}{5}\Big) + n\Big(2 + \frac{1}{5} \times \frac{\frac{5}{10} + 1 \times 10}{5}\Big)$$

$F'_{évaluation} = ln\Big(\frac{34}{50}\Big) + ln\Big(\frac{121}{50}\Big)$
$F'_{évaluation} = 0,497337519$

$$F_{évaluation} = 4,97337519$$

Alors que l'individu : | 1 | 0 | 0 | 1 | 0 |
$S = \frac{3}{5}$, Nbre d'attributs sélectionnés significatifs $= 2$

$$F'_{évaluation} = ln\Big(2 + (1 - \frac{3}{5}) \times \frac{\frac{5}{10} - 2 \times 10}{5}\Big) + ln\Big(2 + \frac{3}{5} \times \frac{\frac{5}{10} + 2 \times 10}{5}\Big)$$

$F'_{évaluation} = ln\Big(\frac{190}{250}\Big) + ln\Big(\frac{223}{50}\Big)$
$F'_{évaluation} = 1,221148766$

$$F_{évaluation} = 12,21148766$$

Dans le chapitre 2.3, nous avons détaillé les différents mécanismes de sélection existants. Nous avons implémenté une sélection en tournoi binaire.

4.2.2 Autres mécanismes de recherche

L'algorithme de base présenté précédemment, dédié à la problématique grâce à des opérateurs et une fonction d'évaluation spécifiques, peut être amélioré pour être plus robuste et donner de meilleurs résultats en explorant mieux l'espace de recherche. Nous proposons diverse améliorations qui interviennent à différents moments du cycle de l'algorithme génétique (voir figure 4.9).

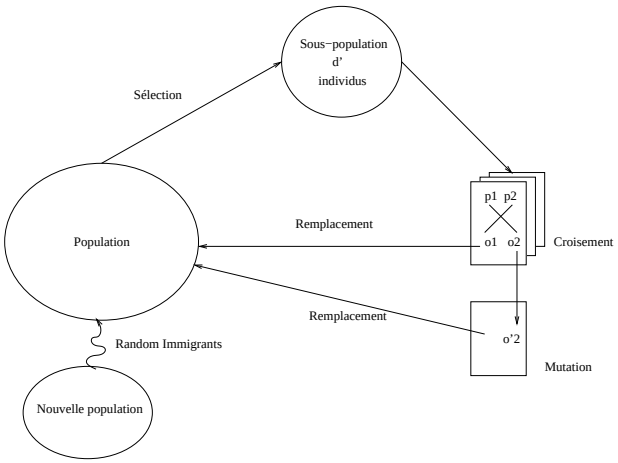


FIG. 4.9 – Les différents étapes de notre algorithme génétique.

4.2.2.1 Elitisme

Pour améliorer les performances de notre algorithme génétique, nous avons décidé d'ajouter une mémoire élite pour préserver les meilleures solutions. On souhaite, en effet, conserver des ensembles d'attributs présentant une fitness que l'on peut considérer comme bonne mais qui seraient perdus au cours des générations.
Soit $\mathcal{E}$ la mémoire élite de taille N ordonnée de façon décroissante en fonction de la fitness des individus. Si, à la génération A(t), on génère un individu a(t) meilleur que $\mathcal{E}(N)$ ($\mathcal{E}(N)$ représentant le moins bon individu de l'élite) et que a(t) n'appartient pas à l'élite $\mathcal{E}$ alors on insère a(t) dans $\mathcal{E}$ si a(t) est à une distance d fixée (en terme de description) des individus de l'élite qui lui sont meilleurs.
Les individus de la mémoire élite interviennent dans la reproduction : en effet une partie des couples générés pour la reproduction sont composés d'un individu de la mémoire élite choisi aléatoirement et d'un individu de la population choisi grâce au procédé de sélection. Dans notre cas, 50% des couples sont composés d'un individu de la mémoire élite.

4.2.2.2 Maintien de la diversité

En général dans un algorithme génétique, la sélection conduit la population vers une distribution uniforme de n copies de l'individu ayant la plus grande fitness.
Nous désirons éviter une convergence prématurée et trouver un certain nombre de combinaisons d'attributs. Pour mettre en balance la pression de sélection, il a fallu mettre en place un niching [HGD94]. Il existe un grand nombre de mécanismes de niching [Pet97], celui implémenté est le "sharing" (partage) introduit par Goldberg et Richardson [GR87].
Le "sharing" accomplit un niching en dégradant la fonction fitness d'un individu en fonction de la présence d'individus similaires. Ce type de sharing requiert donc une notion de distance sur le phénotype ou le génotype de l'individu.
Pour adapter cette notion à notre problématique, nous avons développé une fonction de distance

inter-chromosomique qui correspond au problème dont l'algorithme est donné en 6. Cette fonction compare les deux génotypes des individus dont on veut calculer la distance. Pour cela, elle compare le génotype du premier avec le génotype du second en prenant en compte les gènes présents dans une fenêtre de taille fixe mais s'arrêtant aux coupures chromosomique. La même opération est réalisée avec le second individu qui est comparé au premier. Cette fonction sera réutilisée pour la méthode de clustering. Elle est basée sur la structure des chromosomes car elle prend en compte la structure de l'information qu'elle contient et les coupures qu'ils contiennent. Nous donnons une illustration graphique de la distance dans la figure 4.10.

Algorithme 6 Algorithme de calcul de la distance.

soit σ fixé, soit x et y deux individus
for i=1..nombre d'attributs **do**
  if y_i=1 **then**
    if ∀j in (max(i-σ, précédente coupure chromosomique)..min(i+σ, prochaine coupure chromosomique) $x_j \neq 1$ **then**
      D=D+1
    end if
  end if
  if x_i=1 **then**
    if ∀j in (max(i-σ, précédente coupure chromosomique)..min(i+σ, prochaine coupure chromosomique) $x_j \neq 1$ **then**
      D=D+1
    end if
  end if
end for

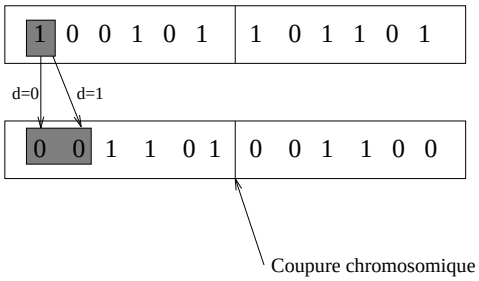


FIG. 4.10 – Calcul d'une distance inter-chromosomique avec d=1. Le résultat est alors D=3.

Une fois le distance définie, il est possible de mettre en place le mécanisme du sharing.
La fonction fitness d'un individu i, $F_{évaluation}(i)$, est dégradée en sommant toutes les "share values" $Sh(d)$ des individus se trouvant dans un rayon fixé σ_{sh} de l'individu i (le compteur de niche m_i) et en divisant $F_{évaluation}(i)$ par cette somme. On a donc :

$$Sh\big(d(I_i, I_j)\big) = \begin{cases} 1 - \left(\frac{d(I_i,I_j)}{\sigma sh}\right)^{\alpha_{sh}} & \text{si } d\big(I_i, I_j\big) < \sigma_{sh} \\ 0 & \text{sinon} \end{cases}$$

α_{sh} et σ_{sh} sont paramétrables. La figure 4.11 montre quelques courbes pour la fonction de sharing

Sh quand σ_{sh} varie. Un coefficient α inférieur à 1 permettra de réaliser une discrimination locale et d'avoir des individus très répartis dans l'espace de recherche.
Le compteur de niche m_i est calculé par :

$$m_i = \sum_{j=1}^{n} Sh\big(d(I_i, I_j)\big)$$

où n est la taille de la population.
Le "sharing fitness" $f_{sh}(i)$ d'un individu i est calculé par :

$$f_{sh}(i) = \frac{F_{évaluation}(i)}{m_i}$$

Les paramètres α_{sh} et σ_{sh} sont fixés en fonction du jeu de données.

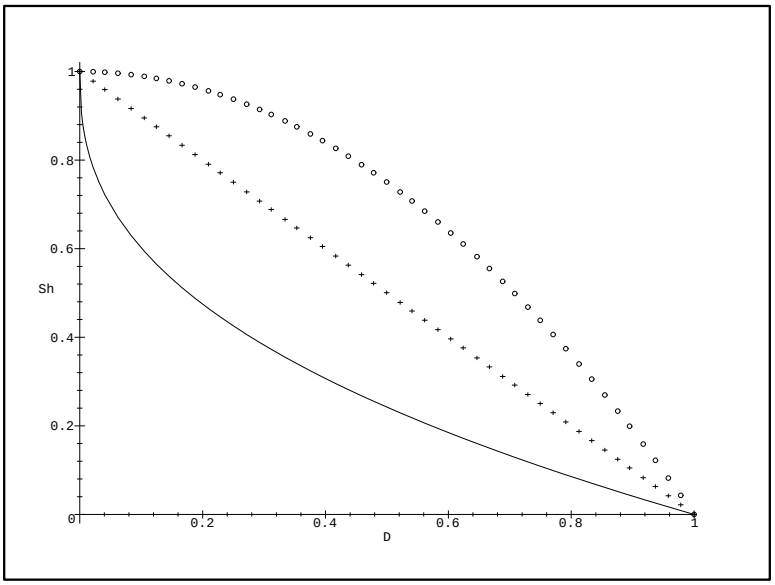


FIG. 4.11 – La fonction de sharing en fonction de $D = \frac{d}{\sigma_{sh}}$ pour α=0.4 (-), α=1 (+), α=2 (o) .

4.2.2.3 Méthode de diversifi cation

Un des défauts que l'on peut remarquer lors de l'évolution de l'algorithme génétique pour ce type de problème est une stagnation de la recherche. Pour remédier à cela, on peut détecter lorsque la recherche stagne : par exemple, le meilleur individu ne s'améliore pas ou la moyenne des notes des individus n'augmente plus pendant un nombre donné d'itérations. On peut alors remplacer une partie de la population par de nouveaux individus générés aléatoirement. Ce remplacement est appelé migration de diversité stochastique ou "Random Immigrant" [Ven97].

Comment ce remplacement se fait-il dans notre cas ?
Tout individu dont la note est inférieure à la moyenne des notes de la population est remplacé par un individu généré aléatoirement. En moyenne, la moitié de la population est remplacée.
Ce remplacement intervient lorsque le meilleur individu n'a pas évolué depuis un nombre de cycles fixé.

4.2.2.4 Evolution dynamique des opérateurs

Il est souvent difficile de choisir la bonne probabilité des opérateurs. Certains auteurs proposent d'adapter les probabilités au caractère dynamique de l'AG. L'idée généralement développée [Ven97] est de mesurer la performance des opérateurs génétiques et d'utiliser cette mesure pour augmenter ou diminuer la probabilité d'application des opérateurs.
Par exemple, plus un opérateur génère d'individus au dessus de la moyenne plus sa probabilité d'application augmente. Ainsi, au début de la recherche, la création de nouveaux individus est favorisée par le croisement puis, lorsque la population a convergé, le taux d'application de la mutation peut augmenter de manière à explorer d'autres régions de l'espace.
Dans notre cas, nous avons évalué expérimentalement le nombre d'individus intéressants que chaque opérateur générait au fur et à mesure des générations. Nous avons remarqué qu'au début de la recherche, la mutation que nous avons mis en œuvre était relativement peu intéressante mais qu'elle apportait de plus en plus d'amélioration sur les individus. Nous avons donc décidé d'augmenter en fonction des expérimentations le taux de mutation au fur et à mesure des générations. Nous verrons par la suite une façon adaptative de gérer les taux d'application des opérateurs.

4.2.2.5 Un mécanisme de mémoire de fréquence

Pour pouvoir mieux parcourir notre espace de recherche, nous avons décidé de mettre en place une mémoire de "fréquence" (cf figure 4.12).
Cette mémoire nous permet de connaître la fréquence de sélection d'un attribut. Elle est mise à jour par les opérateurs de croisement et de mutation ainsi que lors de la génération de la population.
Elle est utilisée lors de la génération de la population du random immigrant : on va en effet générer les nouveaux individus en sélectionnant en priorité des attributs qui ne sont pas fréquemment sélectionnés. Nous avons pour cela sélectionné les attributs à inclure dans les individus générés par le random immigrant en effectuant une roulette inversée par rapport à la matrice de fréquence.

4.2.2.6 Parallélisme en île

Dans le but d'améliorer la robustesse de notre approche et de pouvoir explorer de façon plus intensive l'espace de recherche, nous avons parallélisé notre approche.
Le modèle de parallélisation que nous avons utilisé est celui des îles (island model). Ce modèle est un exemple de Co-évolution hybride de haut niveau [Bac99] à grain de parallélisme moyen. Dans ce modèle, plusieurs algorithmes génétiques s'exécutent de manière parallèle sur des populations de même taille et s'échangent tous les n cycles leurs m meilleurs individus (cf Fig. 4.13). Les individus échangés sont gardés s'ils n'existent pas déjà dans la population de l'algorithme génétique

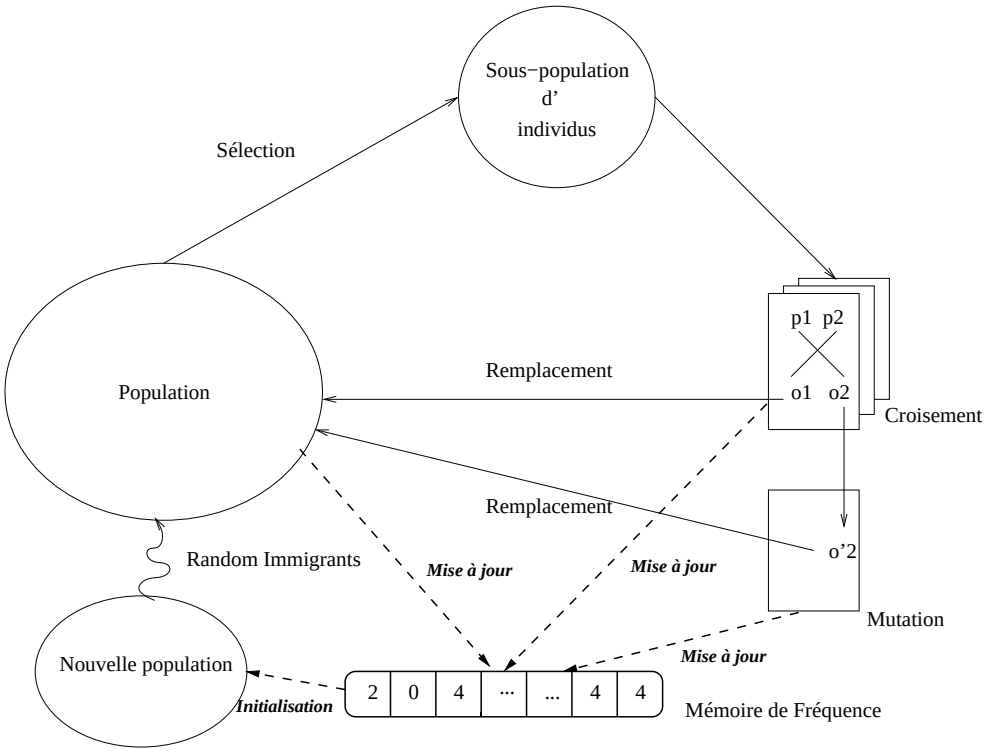


FIG. 4.12 – Illustration du mécanisme de mémoire de fréquence.

recevant et s'ils sont meilleurs que le moins bon des individus. Le voisinage d'un processeur, la fréquence des échanges sont des paramètres de l'algorithme.

Nous donnons ci-dessous la description du processus s'exécutant sur chaque processeur :
- *Répéter*
... Exécution de l'algorithme génétique standard sur la sous-population
faire en parallèle
... Envoi aux processeurs voisins d'individus locaux

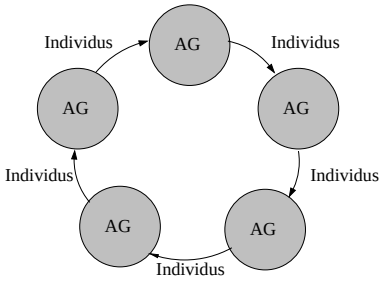


FIG. 4.13 – Illustration du modèle des îles sur une communication en anneau.

Modèle	Meilleur individu	Moyenne Population
AG de base (opérateurs naturels sans doublon)	1,5909	-2,109054
AG dédié	1,9129	1,2387
Ag dédié + RI	1,9129	1,292
AG standard +RI avec mémoire de fréquence	2,0739	1,2941
AG //	2,0739	1,2984

TAB. 4.2 – Tableau récapitulatif des apports des différents mécanisme. RI : Random Immigrant.

... Réception de certains individus voisins
... Remplacement de la sous-population locale
- *Jusqu'à ce que le critère d'arrêt soit vérifi é*

Nous avons choisi de réaliser une communication en anneau : les îles ne transmettent leurs individus qu'à un seul voisin et elles ne reçoivent des individus que de l'île voisine. Les liaisons sont fixées au début de l'exécution. L'algorithme a été implémenté sous l'environnement de programmation parallèle C/PVM. L'implémentation a été réalisée sous Linux et testée sur un réseau de stations hétérogènes

4.2.3 Analyse des différentes méthodes

Nous désirons connaître l'apport des différents mécanismes en terme de qualité de solutions trouvées. Pour cela nous allons comparer les résultats obtenus à l'aide de différentes versions de l'algorithme, comportant plus ou moins de mécanismes de recherche à partir d'une même population initiale générée sur un problème provenant de notre générateur de jeux de données. Le tableau 4.2 nous indique, pour chaque schéma, d'algorithme la valeur du meilleur individu trouvé et la moyenne des notes des individus de la population.
Nous remarquons que le meilleur individu est amélioré par l'introduction d'opérateurs dédiés. En effet, le meilleur individu passe de la note 1,5909 à la note de 1,9129. Nous pouvons donc en conclure que, bien que le codage soit binaire, l'utilisation d'opérateurs naturels n'est pas suffisante pour obtenir de bons résultats. Cela s'observe également sur la note moyenne de la population de l'algorithme génétique en version de base qui est très faible par rapport à la note obtenue dans les autres versions. L'apport des autres mécanismes mis en œuvre (mémoire de fréquence, ...) est visible sur l'amélioration de la meilleure solution. En effet, le meilleur individu obtenu passe de 1,9129 à 2,0739.
Nous donnons quelques courbes d'évolution du meilleur individu et de la moyenne des notes de la population pour quelques schémas implémentés. Les courbes 4.15 et 4.17 nous montrent l'évolution du meilleur individu et de la moyenne des notes sur la population au cours des générations pour l'algorithme génétique dédié à la problématique. Le courbe 4.17 nous montre l'influence du random immigrant sur les notes de la population. Nous pouvons en effet voir apparaître des *drifts* dans la courbe qui correspondent à l'introduction des nouveaux individus générés aléatoirement dans la population.

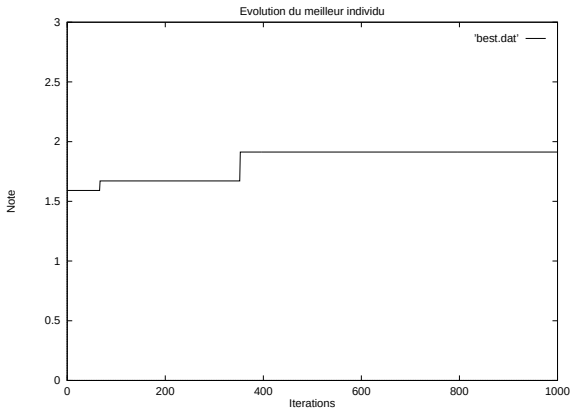


FIG. 4.14 – Evolution du meilleur individu pour le test avec les opérateurs classiques.

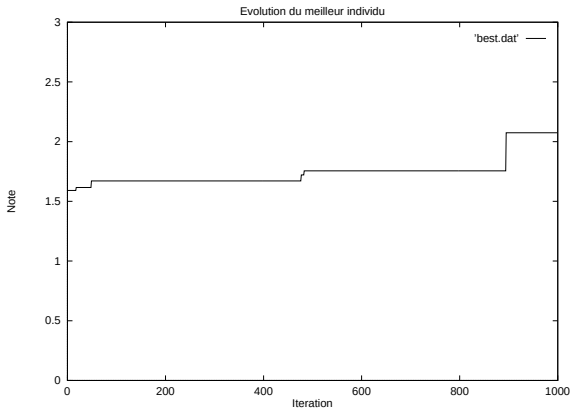


FIG. 4.15 – Evolution du meilleur individu avec les nouveaux opérateurs + Random Immigrant.

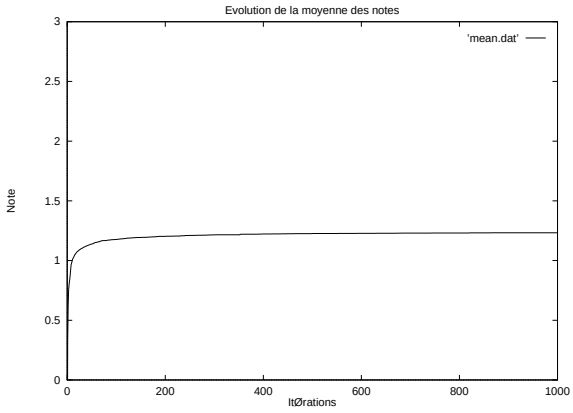


FIG. 4.16 – Evolution des notes de toute la population pour le test avec les opérateurs classiques.

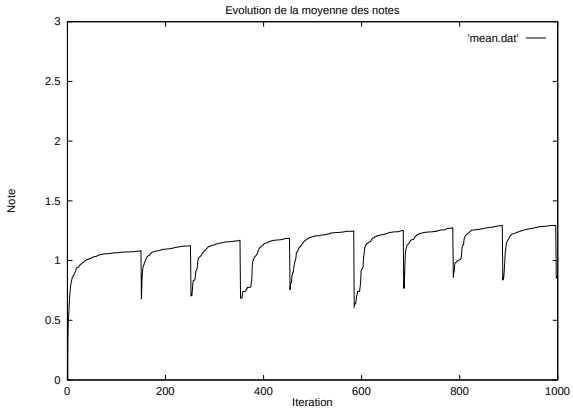


FIG. 4.17 – Evolution des notes de toute la population avec les nouveaux opérateurs + Random Immigrant.

Nous voyons bien à travers ces tests que ces ajouts à notre algorithme génétique améliorent la recherche de bonnes solutions au problème posé.

4.2.4 La seconde phase

L'objectif de cette phase est de valider les attributs sélectionnés, l'algorithme génétique nous donnant déjà des ensembles d'attributs. La seconde phase de notre méthode réalise un clustering à partir des attributs sélectionnés durant la première phase, l'objectif étant de regrouper les paires d'individus partageant les mêmes particularités génétiques.
Deux paires d'individus regroupées dans le même cluster indiquera que ces personnes partagent des particularités pouvant expliquer leur maladie.
La première phase de notre application a sélectionné les attributs les plus significatifs. Nous utilisons maintenant un algorithme classique de la littérature : Kmeans.

Le nombre d'attributs sélectionnés étant relativement petit, nous avons implémenté une version classique de Kmeans en choisissant aléatoirement les centres initiaux. Comme expliqué dans la section 5, l'algorithme Kmeans nécessite qu'on lui indique le nombre de clusters à trouver, or dans notre cas, nous ne le connaissons pas. Nous avons décidé, pour nous permettre de fixer ce nombre, d'utiliser un critère indiquant la qualité de différentes partitions obtenues par des algorithmes de clustering en exécutant plusieurs fois l'algorithme Kmeans avec différentes valeurs de K.
Le *scattering criteria* basé sur des concepts statistiques nous permet de mesurer la qualité du clustering (voir chapitre 5).
La composition des associations est donnée relativement aux centres des clusters. Chaque cluster représente pour nous une association et chaque centre représente la fréquence de chaque attribut dans le cluster. Si un attribut est fréquent dans le cluster alors il appartient à l'association. Lorsqu'il y a des ambiguïtés dans l'affectation d'un attribut f à un cluster C, nous calculons la moyenne de chaque cluster pour cet attribut. Nous gardons en mémoire la valeur minimum pour tous les clusters pour l'attribut considéré. Si la valeur du centre du cluster C moins le minimum trouvé est plus grande que la moyenne alors l'attribut f appartient à l'association représentée par le cluster C.
Cela donne l'algorithme décrit en 7.

Algorithme 7 Algorithme de compositions des associations.

for i=1..nombre d'attributs **do**
  Trouver le $minimum(i)$ sur tous les clusters
  Calculer la moyenne $mean(i)$ sur tous les clusters
end for
for k=1..nombre de clusters **do**
  for i=1..nombre d'attributs **do**
    if $centroid(k_i) - minimum(i) > mean(i)$ **then**
      l'attribut i appartient à k
    end if
  end for
end for

4.3 Résultats

Nous avons réalisé nos expérimentations sur différents jeux de données : les données simulées provenant de GAW11, différents jeux de données provenant de notre générateur de benchmarks et enfin les données réelles issues d'expérimentations réalisées par l'Institut de Biologie de Lille. Nous présenterons également un outil graphique permettant de réaliser les expérimentations et de suivre en temps réel l'évolution de l'algorithme.

4.3.1 Expérimentations sur des données artificielles

Nous avons tout d'abord travaillé sur des fichiers de tests donnés lors du workshop "Genetic Analysis Workshop" de 1998 (GAW11 cf Annexe B) et mis en forme par l'IBL. Les données de

Type bi-point	Marqueurs	multi-points (cM)	multi-points 1/5 (cM)	Commentaire
Chromosome 1 : Locus A	9–10	57–71	11–15	Locus 1 de 2 loci, forme épistatique de la maladie, interagit avec le locus B
Chromosome 1 : Locus D	23–24	170–177	34–35	Allele 1 associé avec la forme à 2 locus. Augmente la chance de maladie
Chromosome 2	N.A	N.A	N.A	
Chromosome 3 : Locus C	45–46	1158–1171	231–235	forme de la maladie à 3 allèles. Interagit avec le facteur d'environnement E1.
Chromosome 4	N.A	N.A.	N.A.	
Chromosome 5 : Locus B	37–38	1867–1878	373–376	Locus 2 de 2 locus, forme épistatique de la maladie, interagit avec le locus A
Chromosome 6	N.A.	N.A.	N.A.	

TAB. 4.3 – Résultats donnés lors de GAW11 [Gre99].

GAW11 simulent une interaction entre 3 gènes et 1 facteur d'environnement dans des familles nucléaires comportant des individus atteints et non atteints. Elles reflètent donc le type de population sélectionnée pour les études de fratries. Ces fichiers ne correspondent pas à des cas réels, mais ils nous permettent de tester nos méthodes car nous connaissons les résultats à trouver.
Pour ces fichiers, nous disposons des paramètres ayant servi à générer les données. Le tableau B.1 donné en annexe nous indique pour chacune des populations les paramètres du simulateur pour les maladies : la fréquence des allèles ou locus et la pénétrance dans la population. Le tableau B.2 lui aussi donné en annexe, indique la taille des fichiers.

Le tableau 4.3 fournit les résultats donnés pour les fichiers du Workshop dans l'article de Greenberg [Gre99]. La figure 4.18 résume graphiquement les différents résultats à obtenir.

Remarques :
Dans l'article [Gre99], D.A. Greenberg explique ce qu'il est facile de découvrir dans les fichiers tests et ce qui l'est moins :

– **Locus C** facile à trouver dans toutes les populations.

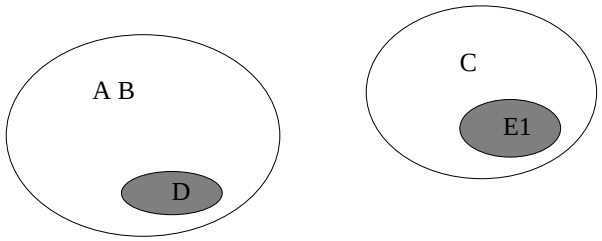


FIG. 4.18 – Récapitulatif des ensembles et sous-ensembles à déterminer pour les fichiers du Workshop GAW11.

Population	Attributs sélectionnés	Support	Fitness	Commentaire
Mycenae	374 374	37 / 92	1,486296	Locus B
	14 348	39 / 100	1,459246	Locus A + Faux positif
	269 338	37 / 97	1,440233	Faux positifs
	36 85	32 / 86	1,419457	Locus D + faux positif
	14 376	32 / 87	1,409953	Locus A + Locus B
	130 148	33 / 90	1,407399	False positives
	12 35	39 / 108	1,395055	Locus A + Locus D
	338 436	33 / 92	1,389688	Faux positifs
	147 236	35 / 98	1,386238	Faux positif + Locus C

TAB. 4.4 – Résultats d'une exécution de l'algorithme génétique pour le jeu de données Mycenae.

Les solutions sont obtenues pour une distance de corrélation σ=7. La légère différence entre le numéro du locus observé et celui à trouver s'explique par cette distance.

- **Locus D** relativement facile à trouver dans les données de Mycenae.
- **Locus B** relativement facile à détecter surtout dans la population de Ruritania.
- **Locus A** difficile à détecter.

Ainsi, l'algorithme génétique a été testé sur cet exemple.
Les paramètres de l'algorithme génétique sont :

- taille de la population : 200 individus
- taux de mutation : 10 %
- taux de sélection : 60 %
- distance de corrélation σ=7
- Nombre de cycles avant random immigrant = 150

Nous appellerons **"faux positif"** un individu issu de la sélection d'attributs par l'algorithme génétique dont la note est bonne et répondant au problème par son support relativement élevé mais qui n'est pas dans les résultats fournis lors du Workshop.

Nous présentons dans le tableau 4.4 le résultat d'une exécution de l'algorithme génétique.

Pour dix exécutions, nous voulons connaître combien de fois une association est découverte par l'algorithme génétique. Le tableau 4.5 présente les résultats obtenus et montre que la première phase de l'approche est capable de découvrir des interactions réelles sur les loci. Certaines d'entre elles étaient plus difficiles que d'autres à découvrir.

Association	A + B	A + D	B + D	C + E1
Fréquence (%)	100	50	20	10

TAB. 4.5 – Associations découvertes par l'AG pour le jeu de données Mycenae.

Nous avons ensuite, dans le cadre de la seconde phase, exécuté plusieurs fois l'algorithme *K*means en utilisant comme seuls attributs ceux sélectionnés par l'algorithme génétique (faux positifs y compris) soit 11 attributs sur les 491 considérés et donc seulement 2,24% du nombre total d'attributs. Cette phase va nous permettre de valider nos résultats.

*K*means va nous aider à découvrir des associations entre les gènes et les facteurs environnementaux. Nous avions tout d'abord voulu tester l'algorithme *K*means sans sélection d'attributs et son temps d'exécution peut être très important (plus de 7500 minutes) et les résultats fournis sont difficilement interprétables (nous ne parvenons pas à déterminer les attributs significatifs). Avec la sélection d'attributs, le temps d'exécution a été réduit à une minute et les résultats sont exploitables. Nous avons testé Kmeans (avec k=2) dix fois sur les 11 attributs sélectionnés. Le tableau 4.6 présente les clusters obtenus et leur nombre d'occurrences. Ce tableau montre que

Cluster	**Occurrences**	**Cluster**	**Occurrences**
(A B D) (E1 C)	4	(A B D) (E1 C D)	1
(A B) (E1 D C)	1	(A B) (E1)	2
(E1 A D B) (E1 C B)	1	(A B D) (E1 D)	1

TAB. 4.6 – Clusters obtenus par *K*means et leurs occurences.

l'algorithme *K*means utilisant les résultats de l'algorithme génétique est capable de construire des clusters relativement proches des solutions présentées lors du workshop.

4.3.2 Expérimentations sur les données générées par les benchmarks

Dans le cadre de la validation de notre approche, nous avons lancé notre méthode sur les données générées par nos soins et avons pour cela proposé un générateur de benchmarks que nous avons présenté précédemment (voir paragraphe 3.2.6 page 39). Nous pouvons paramétrer la taille des benchmarks et nous connaissons les résultats à obtenir.
Nous reportons les résultats pour trois types de benchmarks dont nous présentons les particularités dans le tableau 4.7 (taille, distance de corrélation, ...). Le tableau 4.8 indique pour chaque

Benchmark	# Attributs	AA	NA	NN	Distance de corrélation
ART.IBD.100.20	100	20	30	25	2
ART.IBD.200.50	200	50	0	0	2
ART.IBD.300.60	300	50	0	0	3

TAB. 4.7 – Présentation de quelques benchmarks utilisés par le test. AA représente le nombre de paire d'atteints, NA le nombre de paires d'atteints - non atteints et NN le nombre de paires de non atteint- non atteint.

Benchmark	Solutions générées	Pénétrance(%)	Solutions trouvées	Note
ART.IBD.100.20	11 79	39	10 79	1,636
	37 52 84	32	37 52 84	4,428
ART.IBD.200.50	28 106 113 197	37	28 197	2,777
			106 197	2,666
			113 197	2,5641
ART.IBD.300.60	202 231	42	202 231	2,4606
	79 154 258	40	41 79 154 258	1,1638

TAB. 4.8 – Résulats obtenu sur les jeux de données artificiels par l'algorithme génétique.

benchmark les ensembles d'attributs à découvrir, leur pénétrance et les solutions trouvées par l'algorithme génétique ainsi que leur note.

Nous pouvons remarquer que notre méthode est capable, sur les jeux de données que nous avons générés artificiellement, de retrouver les associations créées par le simulateur. Dans certains cas, ce sont seulement des sous-ensembles de ces associations qui sont trouvés.
Pour permettre une utilisation relativement aisée de notre programme, nous avons réalisé un logiciel dont différentes captures d'écran sont présentées en annexe C.2, C.3, C.3, C.4. Il permet de suivre l'évolution de la recherche de l'algorithme génétique au fur et à mesure des générations comme le montre la capture d'écran 4.19. Cet outil permet de sélectionner le benchmark (ou tout fichier formaté selon la nomenclature utilisée) et de saisir les différents paramètres de l'algorithme génétique et les coefficients de pondération de la fonction d'évaluation. La fenêtre 4.19 donnée par l'onglet temps réel permet de suivre différents indicateurs de l'algorithme génétique. La fenêtre de gauche indique l'évolution au fur et à mesure des générations, la fitness du meilleur individu et la moyenne sur la population. Les deux fenêtres de droite sont spécifiques au problème. La fenêtre supérieure droite indique les attributs sélectionnés par les dix meilleurs individus de l'algorithme génétique et celle inférieure sur toute la population.

4.3.3 Expérimentations sur les données réelles

Des expérimentations sont conjointement menées avec l'Institut de Biologie de Lille sur des données réelles.
Description des données :

– Etude du diabète sur 3 populations :

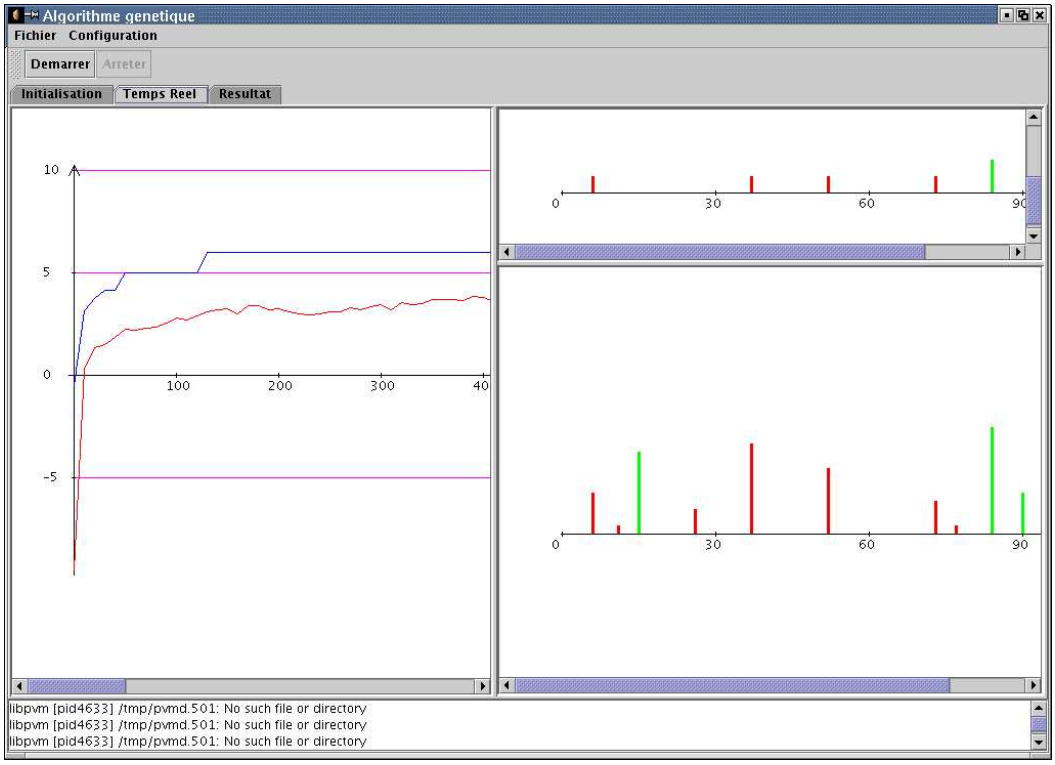


FIG. 4.19 – Fin de convergence de l'algorithme génétique.

  - française,
  - japonaise,
  - mauricienne.
- Etude l'obésité sur 5 populations :
  - obésité massive des adultes,
  - obésité des enfants,
  - les 3 populations du diabète car elles contiennent des obèses.

La nomenclature utilisée est la suivante :
étude/modèle/résolution/essai

Par exemple 1GS/0.2/10 indique :
1GS = diabète sur la population française
0.2 = modèle génétique avec un seuil de "ressemblance génétique" à 0.2
10 = résolution à 10 cM

Les tests ont été effectués avec trois modèles génétiques (0.2, 0.3 et 0.4), avec trois niveaux de résolution (10, 5 et 2 cM) en adaptant les paramètres de l'algorithme génétique et avec 10 essais. Ces différents paramètres génèrent des jeux de données plus ou moins importants en taille comme nous l'indique le tableau 4.9.

Modèle	**# Attributs**	**AA**	**NN**	**NA**
1GS/0.2/10	346	451	51	293
1GS/0.2/5	740	451	51	293
1GS/0.2/2	1834	451	51	293
1GS/0.2/1	3654	451	51	293
Jap	361	208	0	1
MOb/0.4/10	378	767	805	920
MOb/0.7/10	378	767	805	920

TAB. 4.9 – Descriptif de quelques jeux de données utilisés par l'IBL. AA=Atteint Atteint, NN=Non Atteint Non Atteint, NA=Non Atteint Atteint.

Performance de l'algorithme

Notre méthode doit être capable de traiter de grandes bases de données, nous avons donc voulu connaître ses performances. Nous avons pour cela mis en place des expérimentations permettant de déterminer les limites de notre application. L'algorithme génétique représente la phase la plus longue de notre approche et c'est donc sur cette partie que vont s'effectuer nos tests.
Nous avons, dans un premier temps, testé l'algorithme avec différents nombres d'attributs. Les résultats obtenus sont présentés au tableau 4.10. Nous pouvons remarquer que le temps de calcul de l'algorithme génétique est linéaire en fonction du nombre d'attributs.
Dans un second temps, nous avons testé l'algorithme pour des nombres différents d'individus. Les résultats obtenus sont présentés au tableau 4.11.

Résultats

Nous allons nous attacher à décrire les résultats obtenus pour un jeu de données 1GS/0.2/10 (451 paires et 346 attributs). Dans le cadre de ce travail, nous ne prenons en compte que les Atteints/Atteints du tableau 4.9. Nous présentons, tout d'abord, les résultats obtenus par dix exécutions de l'algorithme génétique. Ces exécutions nous ont permis de sélectionner sept attributs correspondant à des emplacements sur les différents chromosomes (A, B, C, D, F, G, H) et un facteur d'environnement ($E1$). Le tableau 4.12 indique la répartition de ces associations.

Pour permettre de valider les résultats et de proposer des associations complètes et intéressantes, nous avons lancé l'algorithme Kmeans avec les huit attributs sélectionnés.
Comme nous ne connaissons pas au préalable le nombre de clusters (i.e. combien d'associations doivent être trouvées), nous avons lancé Kmeans avec plusieurs valeurs de K. Les meilleurs résultats pour le critère utilisé (scattering criteria présenté section 5.4) ont été trouvés pour $k = 3$. Le tableau 4.13 indiquent les nombres de fois où chaque association a été trouvée pour $k = 3$.

	Nombre d'attributs			
	1000	2000	3000	3654
Temps (minutes)	60	140	207	255

TAB. 4.10 – Evolution de la performance en temps en fonction du nombre d'attributs.

	Nombre de paires			
	200	400	600	800
Temps (minutes)	56	64	74	76

TAB. 4.11 – Evolution de la performance en temps en fonction du nombre de paires d'individus.

Association	**B+C**	**D+G**	**A+F**	**A+D**	**E1+H**
Fréquence %	50	100	50	40	10

TAB. 4.12 – Associations découverte par l'AG.

Clusters	**Occurrence**	**Clusters**	**Occurrence**
(A F H)		(A H)	
(B C D)	75%	(B C D)	25%
(E1 G)		(E1 F G)	

TAB. 4.13 – Clusters obtenus et leur occurence.

4.3.4 Conclusion de l'étude des paires de germains

Dans le cadre de ce travail, nous avons mis au point une méthode en deux phases permettant, dans un premier temps, de sélectionner les attributs jugés pertinents et ensuite de réaliser un partitionnement des individus participants à l'étude suivant les attributs sélectionnés grâce à l'algorithme Kmeans. Ce partitionnement nous sert pour donner les associations de facteurs ayant un rôle dans les maladies étudiées.
Pour la première phase de cette approche, nous avons utilisé un algorithme génétique. Pour pouvoir le mettre en œuvre, nous avons tout d'abord réfléchi à la représentation d'un individu. Nous avons choisi la représentation la plus naturelle ce qui nous a conduit à un codage binaire de longueur fixe. Un tel codage permet l'utilisation d'opérateurs de croisement et de mutation "standards" introduits par Goldberg. Nous avons ensuite réalisé des opérateurs adaptés au problème de sélection d'attributs et les avons comparés aux opérateurs standards. Nous avons remarqué que bien qu'il soit plus simple de se contenter d'opérateurs classiques comme le croisement deux points ou la mutation d'un bit, les opérateurs dédiés sont plus performants en terme de qualité de solutions trouvées.
Nous avons ensuite amélioré le parcours de l'espace de recherche de notre algorithme génétique en le dotant de mécanismes avancés permettant une meilleure diversification tels que la mémoire de fréquence, l'élitisme, la migration de diversité stochastique (random immigrant), le parallélisme en îles, ... Ces différents mécanismes nous ont permis d'améliorer la meilleure solution obtenue par rapport à l'algorithme génétique dédié.
Nous avons ensuite testé notre approche sur différents jeux de données artificiels ou réels. Nous avons pu observer que grâce à notre méthodologie, nous sommes capables de fournir des hypothèses correspondantes, pour les jeux de données artificiels, à celles qui ont servi à leur construction.

Nous ne sommes actuellement que dans les phases de production d'hypothèses grâce à cette méthode. Nos partenaires biologistes devront, dans un second temps, réaliser différentes expérimentations pour valider ces hypothèses mais cela prend du temps.

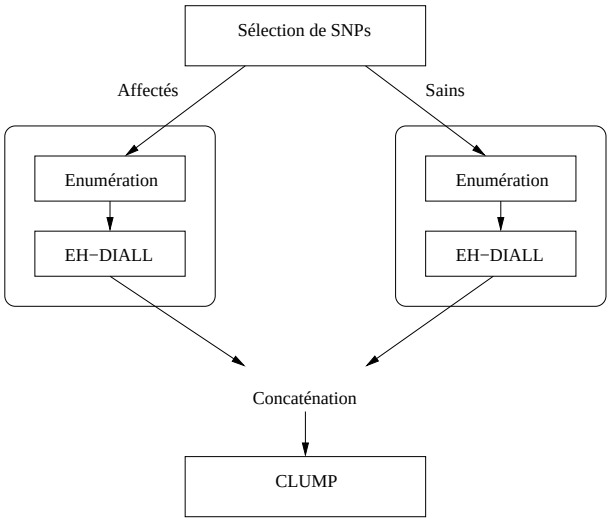


FIG. 4.20 – Processus d'évaluation d'un haplotype.

4.4 Cas de l'étude du déséquilibre de liaison

Dans le cadre de l'étude sur le déséquilibre de liaison présentée dans le chapitre 3.3, nous recherchons des associations de SNPs. Pour cela, nous avons testé une première approche similaire à une sélection d'attributs maximisant le critère de mesure de qualité d'un haplotype donné par les biologistes. Le travail présenté dans cette partie a été initié dans le cadre du stage de DEA de G.Vermeersch [Ver01]. Puis, nous l'avons continué dans le cadre de cette thèse. Dans un premier temps, nous montrerons les particularités de la fonction d'évaluation que nous avons à considérer en terme de paysage et de temps de calcul.
Cette fonction d'évaluation possède des particularités contraignantes nous obligeant à traiter ce problème à l'aide d'un algorithme génétique partitionné en sous-populations.
A partir de ces observations et de ce choix, nous avons développé un algorithme spécifique que nous allons présenter. Nous insisterons sur les adaptations faites pour pouvoir prendre en compte les sous-populations dans les opérateurs. Nous détaillerons les mécanismes adaptatifs mis en place permettant de gérer notamment nos opérateurs qui travaillent sur différentes sous-populations. Ce travail a fait l'objet d'un article présenté à la conférence Evobio 2003 [JDT03a].

4.4.1 Présentation des particularités du problème

Comme nous l'avons vu dans le chapitre 3.3, nous recherchons des combinaisons de SNPs (haplotypes) maximisant une évaluation dont les principes sont donnés par les biologistes. Cette évaluation est très particulière et elle est composée de différents algorithmes qui ont été présentés au paragraphe 3.3.3. Nous allons, tout d'abord, étudier ce processus d'évaluation puis réaliser une étude de la structure du problème en réalisant une énumération exhaustive dans le but d'arriver à déterminer le type de paysage auquel nous sommes confrontés.

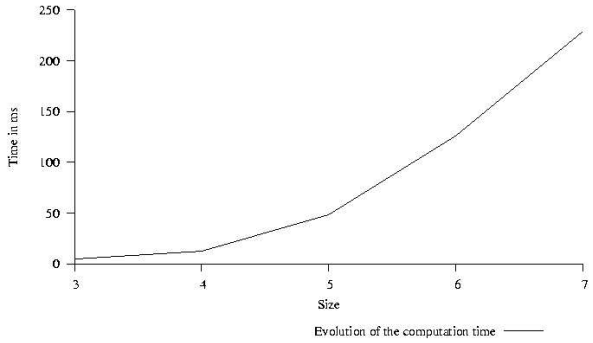


FIG. 4.21 – Temps moyen de l'évaluation d'un haplotype en fonction de sa taille.

4.4.1.1 Le processus d'évaluation

La figure 4.20 représente le processus entier d'évaluation pour un haplotype. À partir d'un ensemble de SNPs candidats, le processus estime d'abord indépendamment, pour les personnes affectées et non affectées, la distribution des allèles dans l'haplotype grâce à EH-DIALL. Puis, CLUMP évalue les associations haplotype-maladie.
Le processus d'évaluation permet de mesurer en termes biologiques la qualité d'un haplotype. L'objectif de l'application est de trouver les haplotypes qui maximisent cette qualité.
Par conséquent, ce problème peut être considéré comme un problème d'optimisation, où l'espace de recherche se compose de toutes les associations possibles de SNPs et où le critère à optimiser est le résultat de l'évaluation. Ce processus d'évaluation peut être très rapide pour des haplotypes composés de deux ou trois SNPs, mais devient plus coûteux quand le nombre de SNPs augmente. Par exemple, sur un Pentium IV 1,7 gigahertz avec 256 MO sur un Linux Redhat 8.0 avec GCC 3.2, un haplotype de taille 3 est évalué en un temps moyen de 6 ms tandis qu'un haplotype de taille 7 est évalué en 201 ms. La figure 4.21 montre l'évolution de la durée de calcul pour l'évaluation des associations de différentes tailles. Nous remarquons que la durée de calcul croît exponentiellement en fonction de la taille de l'haplotype. Cette fonction représente donc un point critique de l'algorithme de recherche.

4.4.1.2 Étude de la structure du problème

Afin de choisir la méthode à employer, nous avons étudié la structure du problème (paysage). Les données actuellement disponibles contiennent 249 SNPs pour 176 individus mais les illustrations sont données pour un problème avec 51 SNPs, car l'espace de recherche des problèmes avec 249 SNPs est trop grand pour être complètement étudié.
Le tableau 4.14 indique le nombre d'haplotypes possibles de différentes tailles pour des problèmes avec 51, 150 et 249 SNPs. Il montre que l'espace de recherche est très grand et impossible à explo-

rer de façon exhaustive. Cette remarque rend l'utilisation d'une énumération complète impossible.

Taille de l'haplotype	**Nombre de solutions** $C_{\#SNP}^{\text{taille haplotype}}$		
	51 SNPs	**150 SNPs**	**249 SNPs**
2	1 275	11 175	30876
3	20 825	551 300	2 542 124
4	249 900	20 260 275	156 340 626
5	2 349 060	591 600 030	7 660 690 674
6	18 009 460	$14,3 \times 10^9$	$3,11535 \times 10^{11}$

TAB. 4.14 – Taille de l'espace de recherche.

Pour étudier la structure du problème, nous avons énuméré toutes les associations possibles avec un nombre restreint de SNPs (associations de 2, de 3 et de 4 SNPs) pour un problème avec 51 SNPs. Pour chaque association, la qualité (résultat des évaluations par EH-DIALL et CLUMP) a été calculée. Les tableaux 4.15, 4.16 et 4.17 montrent les dix meilleurs haplotypes trouvés pour les tailles 2, 3 et 4. Ils donnent la composition des haplotypes et leur qualité (valeur du processus d'évaluation). De plus grandes tables ont été étudiées mais ici seuls les dix meilleurs haplotypes sont présentés.

Haplotype		**Valeur**
$\mathbf{SNP_1}$	$\mathbf{SNP_2}$	
12	33	17,16
12	32	16,28
6	24	16,00
6	31	14,39
6	30	13,97
15	28	13,48
6	27	13,43
21	45	13,30
6	48	13,21
29	38	12,91

TAB. 4.15 – Meilleurs haplotypes de taille 2 (problème de 51 SNPs).

Ces tableaux nous montrent les différentes caractéristiques du problème :

– Nous pouvons tout d'abord remarquer que certains des meilleurs haplotypes de taille n ne sont pas toujours composés des meilleurs haplotypes de taille plus petite. Par exemple, le meilleur haplotype de taille 3 (8-12-15) est composé des couples (8-12), (8-15) et (12-15) qui ont tous un faible score. Comme autre exemple, le meilleur haplotype de taille 4 (8-18-26-50) contient les SNPs 18 et 26 qui ne sont pas présents dans les vingt meilleurs haplotypes de taille plus petite (de taille 2 ou 3). De plus, le SNP 50 est trouvé une seule

Haplotype			Valeur
SNP_1	SNP_2	SNP_3	
8	12	15	58,81
12	16	43	35,35
12	16	38	34,59
12	28	37	33,47
8	32	50	33,10
6	8	16	32,89
12	16	37	32,74
12	31	33	32,55
8	12	33	32,55
12	33	45	32,18

TAB. 4.16 – Meilleurs haplotypes de taille 3 (problème de 51 SNPs).

Haplotype				Valeur
SNP_1	SNP_2	SNP_3	SNP_4	
8	18	26	50	84,85
6	8	16	31	80,63
8	12	15	22	77,21
8	12	15	29	76,68
8	12	15	50	75,80
8	18	26	47	75,73
8	12	15	32	75,61
12	16	33	43	75,56
6	8	16	30	74,39
6	8	12	15	74,33

TAB. 4.17 – Meilleurs haplotypes de taille 4 (problème de 51 SNPs).

Taille haplotype	Min	Max	Moyenne	Écart Type
2	7.10^{-4}	17,16	3,294	2,537
3	2.10^{-2}	58,81	7,543	4,632
4	0,166	84,85	13,899	7,608

TAB. 4.18 – Statistiques sur les solutions (problème de 51 SNPs).

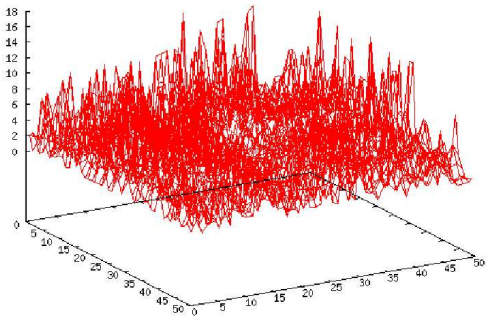


FIG. 4.22 – Paysage obtenu avec des haplotypes de taille 3 (problème de 51 SNPs).

fois parmi les meilleurs haplotypes de taille 3. Cette caractéristique rend l'utilisation de méthodes constructives (greedy algorithm) difficile car ce type d'algorithme essaierait de combiner les bons haplotypes de taille $n-1$ pour obtenir les haplotypes de taille n. Ce type de méthode ne nous permettrait pas d'obtenir tout les bons haplotypes de taille n.

– Les haplotypes de tailles différentes ne sont pas comparables entre eux car les valeurs de leur fonction objective n'ont pas le même ordre de grandeur. En effet, nous pouvons observer que, plus l'haplotype est grand, plus la valeur de sa fonction objective est élevée. Cette observation élimine les algorithmes classiques d'énumération car ils vont construire les haplotypes de grande taille plutôt que des haplotypes de petite taille.

Afin de pouvoir se faire une idée de la répartition des haplotypes de bonne qualité dans l'espace de recherche, nous avons représenté le paysage du problème. Le paysage est donné par l'ensemble des valeurs que peut prendre la fonction objectif pour l'ensemble des haplotypes d'une taille donnée.

La figure 4.22 représente les haplotypes de taille 3 et la valeur de la fonction objective associée. Pour pouvoir obtenir une représentation en 3 dimensions, nous avons choisi aléatoirement un SNP et projeté les valeurs de la fonction objectif. Le choix de ce SNP n'a aucune influence sur le type de paysage obtenu. La figure montre que le paysage est de type plat et rugueux. Ce type de paysage est l'un des plus difficiles à traiter. Avec ce type de paysage, il est important de développer des méthodes ayant un fort pouvoir exploratoire.

Ainsi, l'étude de la structure de ce problème élimine certaines approches de type optimisation. Nous avons montré qu'il était important d'utiliser une méthode étant capable de traiter un espace de recherche relativement grand et ayant un fort pouvoir exploratoire. Nous avons pris en compte ces remarques dans la conception de l'algorithme génétique. Nous détaillons l'algorithme dans la suite de ce chapitre.

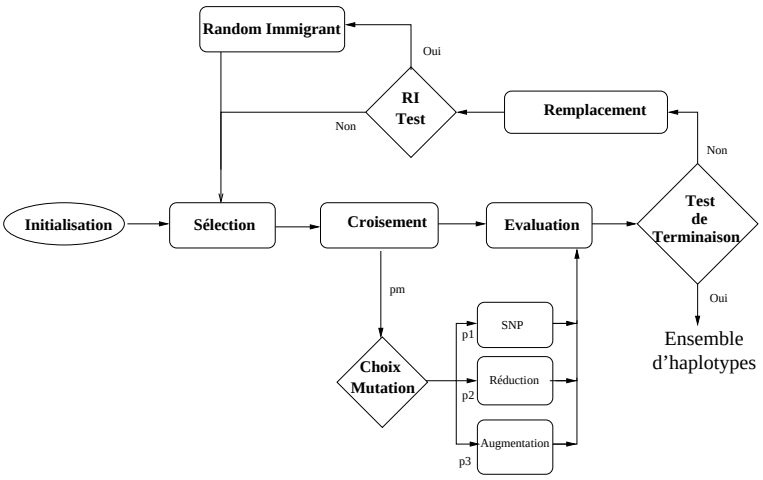


FIG. 4.23 – Schéma général de l'algorithme génétique utilisé.

4.4.2 Un algorithme génétique spécifique

L'objectif de ce travail est de trouver la ou les meilleurs combinaisons de SNPs répondant au problème car nous désirons proposer aux biologistes plusieurs haplotypes. Dans le cadre de cette étude, les biologistes nous ont fourni la fonction qu'ils voulaient optimiser. Après l'étude de ce processus d'évaluation, nous avons développé un algorithme génétique particulier (voir Figure 4.23) qui prend en compte les observations faites précédemment sur la structure du problème. Nous présentons dans cette section les principales caractéristiques et adaptations que nous avons dû effectuer pour pouvoir traiter ce problème dans le cadre d'un problème d'optimisation, en insistant notamment, sur les particularités que la modélisation en sous-populations nous impose de faire, dans le cas des opérateurs et lors de la mise en œuvre d'un mécanisme adaptatif du taux d'application de ces opérateurs.

4.4.2.1 Codage

Un individu code un haplotype d'une taille t donnée. L'haplotype est représenté par un tableau contenant les t SNPs ordonnés dans l'ordre croissant sans répétition.

4.4.2.2 Population et individus

Un des principaux inconvénients de ce problème provient du fait que l'on ne peut pas directement comparer des haplotypes de taille différente. Pour surmonter cela, notre population est subdivisée en sous-populations contenant chacune des haplotypes d'une taille donnée (voir figure 4.24). Le nombre d'individus contenus dans la sous-population est corrélé à la taille de l'espace de recherche associé de manière à suivre l'augmentation de sa taille. Par le biais de certains opérateurs,

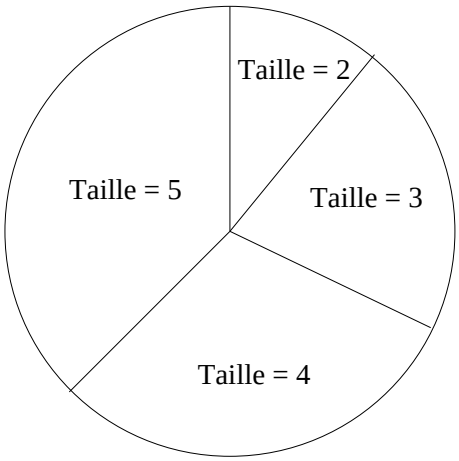


FIG. 4.24 – Exemple de répartition des sous-populations.

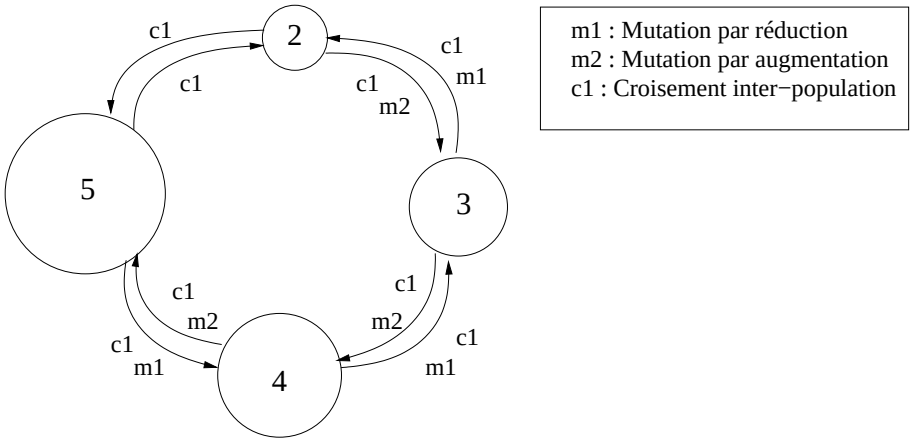


FIG. 4.25 – Coopération entre les sous-populations par différents opérateurs.

il existe des coopérations entre les sous-populations (voir figure 4.25) ce qui permet de partager les SNPs se retrouvant souvent dans les meilleurs haplotypes.

4.4.2.3 Opérateurs

Notre population est divisée en sous-populations contenant des haplotypes ayant un nombre de SNPs fixe au sein d'une même sous-population mais différente entre les sous-populations. Les opérateurs que nous avons mis en œuvre ont été adaptés au codage de l'individu pour permettre de répondre à l'objectif que nous nous sommes fixés mais aussi ils ont aussi été adaptés pour réaliser des coopérations entre les sous-populations.

De plus, comme nous mettons en œuvre plusieurs opérateurs de mutation et de croisement, nous présenterons de façon détaillée le mécanisme adaptatif permettant de fixer les taux d'application des opérateurs.

4.4.2.3.1 Mutation

Nous utilisons trois types de mutation :

- Mutation d'un SNP : un SNP est choisi aléatoirement et remplacé par un autre choisi aléatoirement. Ce processus est similaire à une recherche locale permettant de parcourir le voisinage d'une solution. Nous utilisons cette mutation plusieurs fois en parallèle et nous gardons le meilleur individu généré.
- Mutation par réduction : Un SNP de l'individu est choisi aléatoirement et supprimé de l'haplotype. L'individu est alors de plus petite taille et cela provoque une migration entre les sous-populations.
- Mutation par augmentation : un SNP est ajouté aléatoirement à l'individu. L'individu est alors de plus grande taille et cela provoque également une migration entre les sous-populations.

Ainsi, nous avons trois opérateurs de mutation : mutation d'un SNP, mutation par réduction, mutation par augmentation. Fixer les probabilités d'application des mutations lorsque l'on dispose de plusieurs opérateurs est difficile et, est souvent réalisé de façon expérimentale. Plus un opérateur est efficace plus il sera utilisé. De nombreux auteurs ont mené des expérimentations sur l'adaptation des probabilités d'application des opérateurs [Dav89, Jul95, HL96]. Dans [HWC00], Hong et al. proposent de calculer ce nouveau taux de mutation en évaluant le progrès d'une mutation M_i pour un individu ind muté en un individu mut (nous réutiliserons ce principe à la section 6.4.6) par :

$$progress(M_i) = Max\big(fitness(ind), fitness(mut)\big) - fitness(ind)$$

L'inconvénient de l'approche que nous avons utilisée dans notre cas est que l'on peut obtenir par mutation des individus de taille différente et donc de fitness non comparable. Nous avons donc décidé, pour mettre en place une approche de ce type, de normaliser le progrès à l'aide du meilleur et du plus mauvais individu de la sous-population correspondante à l'individu mut. Nous définissons alors la valeur normalisée de la fitness d'un individu ind comme :

$$norm(ind) = \frac{fitness(ind) - worst\_of\_size(ind.size)}{best\_of\_size(ind.size) - worst\_of\_size(ind.size)}$$

Le progrès devient alors :

$$progress_j(M_i) = Max\big(norm(ind), norm(mut)\big) - norm(ind)$$

Pour tous les opérateurs de mutation M_i, soit $Nb\_mut(M_i)$ le nombre d'applications de la mutation à une génération donnée ($j = 1, .., Nb\_mut(M_i)$). Nous pouvons alors calculer le profit de l'opérateur de mutation M_i :

$$Profit(M_i) = \frac{\sum_j progress_j(M_i)/Nb\_mut(M_i)}{\sum_k \Big(\sum_j progress_j(M_k)/Nb\_mut(M_k)\Big)}$$

pour N opérateurs de mutation à appliquer, nous fixons le taux de mutation minimal pour une mutation à δ et le taux maximal à $p_{mutation}$. Le nouveau taux d'application pour chaque opérateur

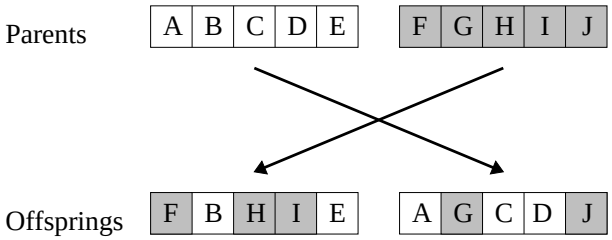


FIG. 4.26 – Croisement uniforme entre individus de taille 5.

de mutation M_i est calculé en utilisant la formule suivante [HWC00] :

$$p(M_i) = Profit(M_i) \times (p_{mutation} - N \times \delta) + \delta$$

La somme de tous les taux d'application des mutations est égal au taux de mutation global $p_{mutation}$. Le taux de mutation initial étant, pour chaque opérateur, fixé à $p_{mutation}/N$.
Ainsi, il n'est pas nécessaire d'étudier au préalable les performances des opérateurs et leur adaptation à chaque type de problème.

4.4.2.3.2 Croisement

Nous utilisons un croisement de type uniforme : à partir des deux chaînes de SNPs des parents, nous créons deux enfants en combinant aléatoirement les variables correspondant au SNP de chaque site (voir figure 4.26). L'un des enfants obtenu est éventuellement envoyé à la mutation, puis la fitness et le nombre de SNPs des enfants sont calculés.

Nous utilisons deux types de croisement :
- Intra-population : seuls les croisements entre individus d'une même sous-population sont autorisés,
- Inter-population : les croisements entre individus de différentes sous-populations sont autorisés. Ce croisement est de type pseudo uniforme. En effet, nous imposons que chacun des enfants aient la même taille que l'un de leurs parents.

Les probabilités d'application de chaque type de croisements sont fixées de façon adaptative. Nous avons adopté la même stratégie que celle développée pour les mutations précédemment au cas du croisement quadratique. Nous définissons l'amélioration moyenne ($Improve_{Intra}$) d'un enfant e par rapport à ses deux parents p_1 et p_2 pour le croisement intra-population comme :

$$Improve_{Intra}(e, p_1, p_2) = \left(\begin{array}{l} \frac{Max(fitness(p_1), fitness(e)) - fitness(p_1)}{best\_of\_size(e.size)} \\ + \\ \frac{Max(fitness(p_2), fitness(e)) - fitness(p_2)}{best\_of\_size(e.size)} \end{array} \right) /2$$

Dans ce cas, les trois individus e, p_1 et p_2 sont de même taille et peuvent donc être comparés.

Nous définissons l'amélioration moyenne ($Improve_{Inter}$) pour le croisement inter-population en comparant uniquement l'amélioration entre un enfant e et son parent de même taille (il en existe toujours un) :

$$Improve_{Inter}(e,p_1,p_2) = \begin{cases} \frac{Max(fitness(p_1),fitness(c))-fitness(p_1)}{best\_of\_size(e.size)} & \text{If } size.p_1 = size.e \\ \frac{Max(fitness(p_2),fitness(c))-fitness(p_2)}{best\_of\_size(e.size)} & \text{If } size.p_2 = size.e \end{cases}$$

Donc la fonction globale de l'amélioration est :

$$Improve(e,p_1,p_2) = Improve_{Intra}(e,p_1,p_2) \;\; OR \;\; Improve_{Inter}(e,p_1,p_2)$$

Le progrès de la $j^{\text{ème}}$ application de chaque croisement C_i qui, à partir de deux individus p_1 et p_2 produit deux enfants e_1 et e_2, est alors :

$$progress_j(C_i) = Improve_j(e_1,p_1,p_2) + Improve_j(e_2,p_1,p_2)$$

Pour tous les opérateurs de croisement C_i, supposons qu'il y ait $Nb\_cross(C_i)$ applications du croisement durant un nombre donné de générations. Le profit de l'opérateur de croisement C_i est alors :

$$Profit(C_i) = \frac{\sum_j progress_j(C_i)/Nb\_cross(C_i)}{\sum_k \left(\sum_j progress_j(C_k)/Nb\_cross(C_k)\right)}$$

Nous posons δ comme taux minimum d'application d'un croisement et $p_{crossover}$ comme taux global d'application du croisement pour N opérateurs de croisement à appliquer. Le nouveau taux d'application de chaque opérateur de croisement C_i est calculé en utilisant la formule suivante [HWC00] :

$$p(C_i) = Profit(C_i) \times (p_{crossover} - N \times \delta) + \delta$$

La somme de tous les taux d'application de croisement est égale au taux global $p_{crossover}$. Le taux initial de chaque croisement est mis à $p_{crossover}/N$.

4.4.2.4 Mécanisme de diversification

L'espace de recherche étant grand, il est important d'avoir une grande capacité d'exploration. Le mécanisme de Random Immigrant permet de maintenir une diversité dans la population en introduisant à certains moments de nouveaux individus [Con95]. Il permet aussi d'éviter une convergence prématurée. Nous utilisons ce mécanisme (voir figure 4.23) lorsque le meilleur individu est le même durant un nombre N de générations. Alors, tous les individus de la population, qui ont une valeur d'évaluation en dessous de la moyenne de la population, sont alors remplacés par de nouveaux individus générés aléatoirement.

4.4.2.5 Parallélisme

Nous avons remarqué pendant nos expérimentations que la fonction d'évaluation utilisée était relativement coûteuse en temps (voir figure 4.21). Pour permettre l'utilisation de notre algorithme

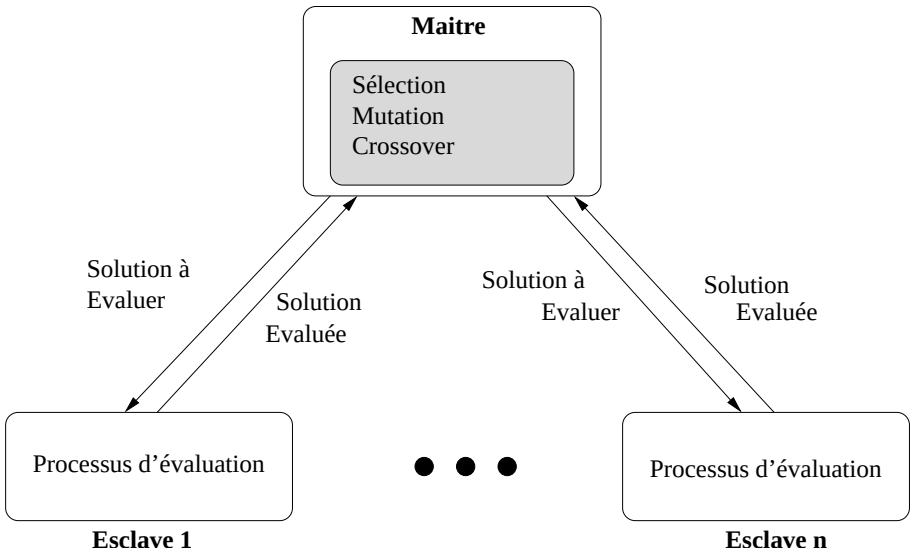


FIG. 4.27 – Modèle synchrone maître / esclave utilisé pour la parallélisation.

en un temps raisonnable, nous avons effectué une implémentation parallèle synchrone de la phase d'évaluation.
Cette implémentation est basée sur un modèle maître / esclave (voir figure 4.27). Les esclaves sont initialisés au début du programme et accèdent seulement une fois aux données. Durant la phase d'évaluation, le maître donne un individu à évaluer par esclave. L'esclave calcule la fitness de l'individu et le renvoie au maître.
Nous avons réalisé cette implémentation en C/PVM (Parallel Virtual Machine) [GBD+94]. L'implémentation a été réalisée sous Linux et testée sur un réseau de stations hétérogènes.

4.4.2.6 Remplacement et test de terminaison

Un nouvel individu est inséré dans la population courante s'il est meilleur que le plus mauvais des individus de la population et s'il n'est pas déjà présent.
Afin de permettre à l'algorithme de progresser, nous avons décidé de l'arrêter lorsque le meilleur individu n'a pas été amélioré depuis un nombre fixé de générations. Ce nombre de générations est supérieur à celui fixé pour le random immigrant.

4.4.3 Expérimentations

Nous avons testé notre méthode sur des données provenant de l'Institut de Biologie de Lille (voir section 3.3). Ce jeu de données contient 106 individus : 53 malades, 53 non malades.

Nous avons testé l'algorithme génétique proposé afin de trouver la meilleure configuration. Nous proposons pour cela de tester les schémas suivants :

- Avec et sans la mutation par réduction et par augmentation.
- Avec et sans random immigrant.

– Avec et sans le croisement inter-population.

Il apparaît que les mécanismes intervenant entre les sous-populations sont efficaces et permettent de trouver de meilleures solutions. L'opérateur de random immigrant est, quant à lui, capable d'introduire de la diversité quand la recherche ne progresse plus.

L'algorithme génétique possède de nombreux paramètres et l'utilisateur peut lui même fixer la taille maximale d'un haplotype. Les biologistes ont choisi de faire des expérimentations avec une taille maximale d'haplotype égale à 6. Nous fixons ensuite :

– $P_{mutation}$ = 0,9
– $\delta = 0.1$
– Taille de la population = 150
– Nombre de générations durant lesquelles le meilleur est le même (test de fin) = 100
– Taille maximale d'un haplotype = 6
– Stagnation du meilleur individu (Random Immigrant) : 20

Le tableau 4.19 présente les résultats obtenus avec les différentes combinaisons de nos mécanismes. Pour chaque combinaison, nous avons effectué dix exécutions. Nous indiquons les mécanismes utilisés dans la colonne "Schéma", le meilleur haplotype trouvé sur les exécutions pour chaque sous-population, sa taille et sa valeur de fitness. Nous indiquons ensuite la moyenne obtenue sur les différentes exécutions et la déviation (Dev.) par rapport au meilleur individu attendu. Enfin, nous donnons le nombre minimum et moyen d'évaluations nécessaires pour obtenir la solution. Nous comparons ce nombre moyen d'évaluations à la taille de l'espace de recherche pour indiquer dans le tableau le pourcentage de l'espace de recherche exploré.

Nous surlignons en gris le meilleur haplotype trouvé lorsque l'algorithme n'a jamais trouvé l'haplotype optimal au cours des différentes exécutions. Pour indication une déviation nulle indique que l'haplotype optimal est trouvé à chaque exécution. L'haplotype optimal a été déterminé pour les tailles 2, 3 et 4 grâce à l'énumération (voir section 4.4.1.2). Pour les tailles 5 et 6, il correspond au meilleur haplotype jamais trouvé lors de toutes les exécutions.
Nous remarquons que lorsque nous travaillons avec un algorithme génétique classique (appelé ici AG simple sans opérateur inter-population), nous trouvons pour les haplotypes de taille 4 une valeur de fitness moyenne de 78,942 sur les exécutions et l'algorithme n'a jamais trouvé l'optimal, alors que le schéma le plus complet mis en oeuvre (Mutation adaptative + Croisement Adaptatif + Random Immigrant) le trouve à chaque fois.
Nous pouvons également observer que l'introduction du mécanisme de Random Immigrant (schéma 2) permet d'améliorer significativement la qualité des solutions obtenues pour les tailles 4 et 6 en introduisant plus de diversité. De plus, nous pouvons remarquer que les opérateurs inter-populations permettent de rendre plus robuste la méthode car les solutions optimales sont obtenues plus régulièrement d'où une déviation plus faible en particulier sur les haplotypes de plus grande taille. Les schémas 4 et 5 montrent que, lorsque l'on fixe adaptivement l'application des opérateurs, l'algorithme est encore plus robuste.
Nous pouvons en conclure, qu'afin d'obtenir le maximum de diversité et de robustesse, nous devons combiner les différents mécanismes présentés précédemment : random immigrant, croisement et mutation inter-populations à taux adaptatifs ce qui correspond au schéma d'expérimentation 6.

TAB. 4.19 – Comparaison des résultats obtenu par l'AG sur le problème de 51 SNPs.

Schéma	Meilleur Haplotype	Fitness	Moyenne	Dev	# min. d'éval.	# moy. d'éval.	% Espace exploré
1. AG simple	8 12 15	58,814	58,814	0	175	782	3,75
sans opérateur	6 8 16 31	80,631	80,29	4,87	1938	2409,6	0,964
inter-population	8 12 16 33 43	123,108	123,108	0	2581	5207,8	0,221
	8 12 15 21 32 43	161.252	158,818	2,36	3762	11153,7	0,061
2. AG simple	8 12 15	58,814	58,814	0	167	577,3	2,77
+ Random Immigrant	8 18 26 50	84,856	82,478	2,92	1287	4156,4	1,66
	8 12 16 33 43	123,108	123,108	0	1375	6166,2	0,26
	8 12 15 21 32 43	161,252	160,77	0,47	6051	10853,6	0,06
3. AG simple	8 12 15	58,814	58,814	0	187	425,8	2,04
avec des opérateurs	6 8 16 31	80,631	79,262	5,38	644	3584	1,43
non adaptifs	8 12 16 33 43	123,108	123,108	0	1778	6749	0,29
inter-populations	8 12 15 21 32 43	161,252	161,252	0	6676	11620	0,06
4. Mutation adaptative	8 12 15	58,814	58,814	0	302	767,9	3,69
+ Croisement intra	8 18 26 50	84,856	81,98	2,92	375	2608,6	1,04
population	8 12 16 33 43	123,108	123,108	0	2397	2745	0,12
	8 12 15 21 32 43	161,252	161,252	0	3364	10275,3	0,06
5. Mutation adaptative	8 12 15	58,814	58,814	0	207	1094,6	5,26
+ Croisement adaptatif	8 18 26 50	84,856	82,82	1,79	426	3419,5	1,37
	8 12 16 33 43	123,108	123,108	0	2227	4846,9	0,21
	8 12 15 21 32 43	161,252	161,252	0	4455	10564	0,06
6. Mutation adaptatif	8 12 15	58,814	58,814	0	317	506,5	2,43
+ Croisement adaptatif	8 18 26 50	84,856	84,856	0	1111	2822,9	1,13
+ Random Immigrant	8 12 16 33 43	123,108	123,108	0	2994	3815,5	0,16
	8 12 15 21 32 43	161,252	161,252	0	11573	13082	0,07

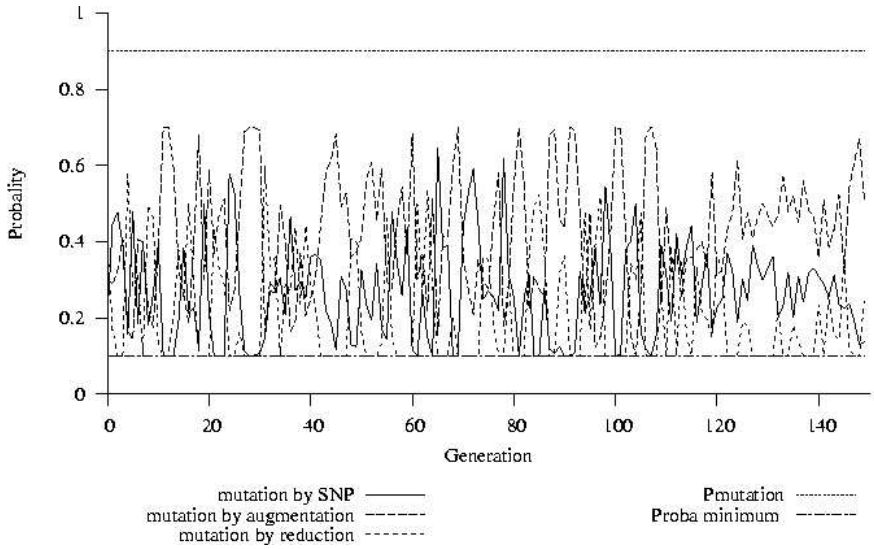


FIG. 4.28 – Exemple d'évolution des taux d'application des opérateurs de mutation.

L'introduction de ces différents mécanismes requiert évidemment un effort de calcul plus important. Nous pouvons remarquer néanmoins que tous les schémas testés explorent relativement peu de solutions par rapport à la taille de l'espace de recherche (voir tableau 4.14).

En complément, nous avons voulu connaître l'intérêt du taux d'application adaptatif des opérateurs. Les figures 4.28 et 4.29 montrent l'évolution des taux de mutation et respectivement de croisement lorsque l'on met en œuvre le mécanisme d'adaptativité. Nous pouvons remarquer, surtout pour la mutation, l'importante modification des taux.
Finalement nous avons étudié la convergence de l'algorithme et les figures 4.30 et 4.31 montrent l'évolution de la recherche durant les générations. Pour chaque sous-population nous indiquons la valeur du meilleur individu ainsi que la valeur de la moyenne de la fitness sur la population. Les deux courbes les plus basses correspondent ainsi au meilleur individu et à la valeur moyenne de la fitness pour les haplotypes de taille 3, les deux suivantes correspondent à un haplotype de taille 4, ... Nous pouvons remarquer que la convergence vers un individu de bonne qualité se fait relativement rapidement. Nous pouvons observer sur la figure 4.31 que le mécanisme de random immigrant est utilisé car des fossés apparaissent sur la courbe de la moyenne de la population. Ceci s'explique par l'introduction de nouveaux individus générés aléatoirement pouvant être de mauvaise qualité, mais introduisant une diversité capable, quelques itérations plus tard, d'améliorer le meilleur individu.

Ainsi, cet algorithme a montré qu'il était capable de fournir des solutions de bonne qualité pour ce problème. Notre première étude a été effectuée sur un sous-ensemble de 51 SNPs provenant du jeu de données final contenant 249 SNPs. Nous proposons une expérimentation sur le jeu complet

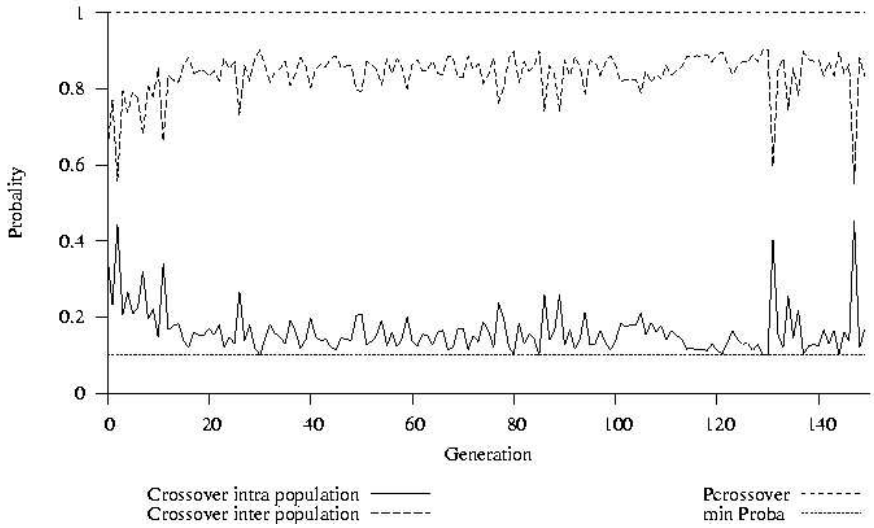


FIG. 4.29 – Exemple d'évolution des taux d'application des opérateurs de croisement.

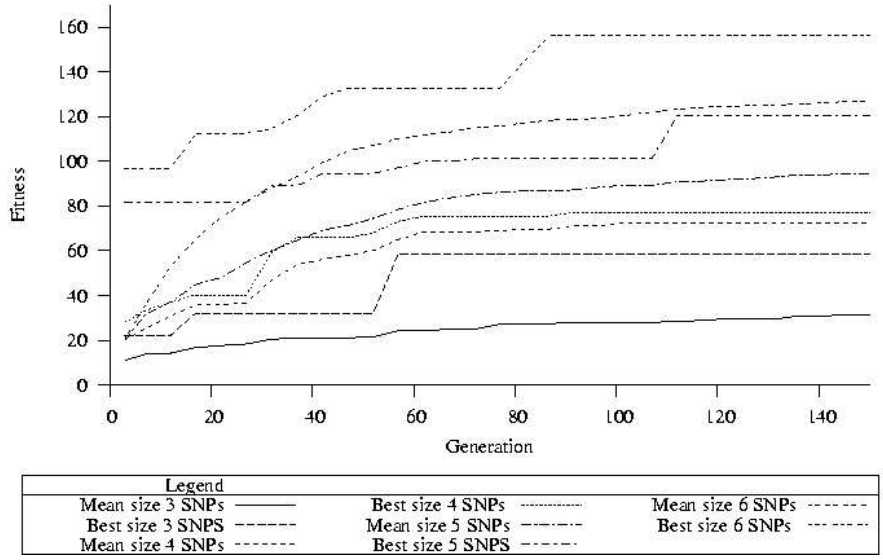


FIG. 4.30 – Exemple d'évolution de la moyenne et du meilleur individu pour des haplotypes de taille 3, 4, 5, 6 sans random immigrant.

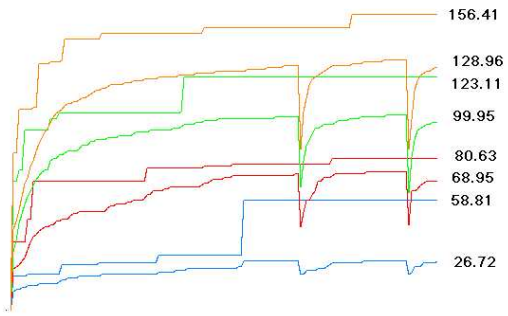


FIG. 4.31 – Exemple d'évolution de la moyenne et du meilleur individu pour des haplotypes de taille 3, 4, 5, 6 avec random immigrant.

Meilleur Haplotype	**Fitness**	**Moyenne**	**Dev**	**# min. d'éval.**	**# moy. d'éval.**	**% Espace exploré**
18 202 231 ou 18 202 230	74,746445	74,746445	0	905	1145	0,045
18 47 196 202	130,971878	130,971878	0	2195	3668	0,0023
18 19 54 95 202	203,321335	199,904397	2,130	5347	8022,4	0,000104
18 45 54 95 202 231	253,057281	251,9672604	2,362	9771	12495	0,000004

TAB. 4.20 – Résultats obtenus par l'AG sur 249 SNPs.

dont les résultats sont donnés au tableau 4.20.

4.4.4 Conclusion sur l'étude du déséquilibre de liaison

Nous avons modélisé le problème de l'étude du déséquilibre de liaison comme un problème de sélection d'attributs pertinents, la pertinence d'un attribut étant mesurée par des critères fixés par les biologistes. Le processus d'évaluation donné par les biologistes est une combinaison de deux programmes statistiques qui permettent de mesurer l'association haplotype-maladie. Après avoir effectué une analyse de cette fonction d'évaluation et du paysage qui lui est associé grâce à une énumération exhaustive des solutions, nous avons pu cerner la structure du problème. Ainsi nous avons développé un algorithme génétique permettant de générer des solutions de différentes tailles grâce à l'utilisation de sous-populations.
Cette particularité nous a amené à travailler sur des opérateurs permettant la coopération entre les sous-populations, aussi bien pour la mutation que pour le croisement. Nous avons donc développé plusieurs opérateurs adaptés au codage de solutions que nous avons adoptés. Pour fixer les taux

d'application de ces opérateurs, nous avons choisi d'utiliser une méthode adaptative que nous avons étendue à notre structure en sous-populations et aux coopérations pouvant avoir lieu entre elles. Une série d'expérimentations nous a montré la robustesse de notre approche et en particulier l'apport des différents mécanismes. De plus, l'énumération exhaustive des haplotypes de taille 2, 3 et 4 que nous avions réalisée pour l'étude de paysage nous a donné les solutions optimales auxquelles nous avons pu nous comparer.
Pour permettre de traiter plus rapidement les jeux de données réels, nous avons également proposé une parallélisation de la fonction d'évaluation dans un schéma maître-esclave.
La validation biologique de ces résultats doit être réalisée par notre partenaire de biologiste.

4.5 Conclusion

Nous avons présenté dans ce chapitre deux méthodes permettant d'extraire des attributs pertinents pour les problématiques auxquelles nous étions confrontés : l'étude des paires de germains et l'étude par déséquilibre de liaison. Ces méthodes sont basées sur des métaheuristiques à base de population de solutions, les algorithmes génétiques, qui nous ont demandé un travail de modélisation des problèmes en problème d'optimisation aussi bien pour la fonction objective que pour le codage une solution (un individu). Chacun de ces travaux montre que nous avons su prendre en compte des connaissances du domaine afin de développer les méthodes les mieux adaptées.
Ces approches nous ont permis de remarquer l'importance de l'utilisation d'opérateurs génétiques dédiés à la problématique même lorsque l'emploi d'opérateurs naturels est possible. Nous avons également mis en œuvre des opérateurs génétiques dont les performances varient au cours de la recherche. Nous avons étudié leur apport et mis en place dans les deux méthodes une gestion automatique de leur taux d'application.
La nature différente de ces approches nous a permis d'aborder deux types de parallélisme. Le premier type est un modèle distribué basé sur une approche en île favorisant la coopération. Le second type est un parallélisme centralisé de la fonction d'évaluation qui permet de traiter massivement la tâche critique.

Chapitre 5

Méthodes d'optimisation pour le clustering

La catégorisation, ou clustering, consiste à trouver des groupes homogènes au sein d'une population. Le partitionnement est une tâche d'apprentissage "non supervisée" car on ne dispose d'aucune autre information préalable que la description des exemples. Le clustering a été utilisé dans de nombreux contextes et par de nombreux chercheurs dans différentes disciplines en tant que processus d'analyse exploratoire de données.
Dans ce chapitre, nous présentons une approche par méthode évolutionnaire pour traiter le problème de clustering. La particularité de cette méthode provient de plusieurs aspects. Tout d'abord nous voulions réaliser un algorithme déterminant de lui même le nombre de clusters réalisant une meilleure partition des données. A partir de cet objectif, nous avons déterminé un codage des solutions permettant de prendre en compte la taille variable d'une solution en nombre de clusters. Nous avons ensuite déterminé des opérateurs adaptés au problème et au codage choisi. Pour permettre une meilleure efficacité de notre algorithme, nous avons fait une hybridation avec une itération de l'algorithme Kmeans. Dans ce cadre, nous avons travaillé sur une nouvelle définition de centre de cluster permettant de travailler notamment avec des données nominales. Nous avons associé à cette nouvelle définition une nouvelle distance dont nous démontrerons les propriétés. Afin de permettre d'évaluer les résultats obtenus par l'algorithme génétique que nous proposons, nous nous comparerons sur quelques jeux de données à d'autres algorithmes classiques de la littérature.
Nous décrirons, dans un premier temps, la tâche d'extraction de connaissances considérée : le clustering et les algorithmes classiques de la littérature qui nous servirons pour les comparaisons des performances puis nous présenterons la méthode que nous avons développée, CHyGA, et les résultats obtenus.

5.1 Le clustering : généralités

Dans cette partie, nous donnerons le formalisme général d'une tâche de clustering, puis nous décrirons quelques méthodes classiques de clustering.

Dans le modèle d'apprentissage non supervisé, on ne connaît pas a priori de classes. Les mo-

dèles sont obtenus automatiquement par analyse des régularités dans les données. L'apprentissage non supervisé peut se caractériser de la manière suivante :
Étant donné un ensemble d'observations $O_1, ..., O_n$ de dimension d, non étiquetées, trouver un regroupement des observations en classes en maximisant des critères du type "similarités intra-classes" et "distances inter-classes".

5.1.1 Composition d'une tâche de clustering

La tâche de clustering implique les étapes suivantes :

- Représentation du modèle de données (avec éventuellement extraction d'attribut et/ou sélection) : elle se rapporte au nombre présumé de clusters, au nombre d'exemples disponibles et au nombre, type et échelle des attributs disponibles pour l'algorithme de clustering. Certaines de ces informations ne sont pas toujours disponibles.
  La sélection d'attribut est un processus qui identifie le sous-ensemble d'attributs le plus efficace parmi les attributs originaux pour réaliser le clustering (voir chapitre 4).
  L'extraction d'attribut représente l'utilisation d'une ou plusieurs transformations des attributs originaux pour obtenir de nouveaux attributs plus pertinents.

- Définition d'une mesure de proximité appropriée au domaine des données : la proximité des exemples est habituellement mesurée par une fonction de distance définie sur des paires d'exemples. Une grande variété de mesures sont utilisées dans les différentes communautés.

- Clustering ou segmentation :cette phase peut être réalisée de nombreuses façons. Les clusters de sorties peuvent être durs (une partition des données dans des groupes) ou floues (fuzzy) où chaque exemple a un degré variable d'appartenance à chaque cluster de sortie.
  Les algorithmes de clustering hiérarchiques produisent une série de partitions basées sur un critère de similarité.
  Les algorithmes de clustering par partition identifient la partition qui optimise (souvent localement) un critère de clustering.

- Abstraction des données (si nécessaire) : c'est un processus d'extraction d'une représentation simple et compacte de l'ensemble de données. Ici, la simplicité est soit orientée dans la perspective d'une analyse automatique qui pourra ainsi être réalisée de manière plus efficace par les logiciels, soit orientée vers l'utilisateur humain afin que la représentation obtenue soit facile à comprendre et intuitive. Dans le cas du clustering, une abstraction typique des données est une description compacte de chaque cluster, en général soit en terme de prototype, soit en terme d'exemples (on donne le centre de chaque cluster,...).

- Évaluation du résultat (si nécessaire) : cette phase analyse la répartition de la sortie de la procédure de clustering. Souvent, cette analyse utilise un critère spécifique d'optimalité ; malgré tout, ces critères sont souvent subjectifs. En fonction du type de clustering, il existe différents types de validation.

5.1.2 Définitions

Après avoir présenté les différentes phases d'une tâche de clustering, nous donnons quelques définitions nécessaire à la compréhension du problème.

Définition 13 *Un pattern (exemple ou vecteur d'attributs ou observation ou objet ou item)* $\mathbf{x}$ *est un article de données utilisé par l'algorithme de clustering. En général, c'est un vecteur de dimension* d *:* $\mathbf{x} = (x_1, ..., x_d)$

Définition 14 *Chaque élément* x_i *du pattern* $\mathbf{x}$ *est appelé "feature" ou attribut.*

Définition 15 d *est la dimension du pattern ou de l'espace des patterns et est égale au cardinal des attributs.*

Définition 16 *Un ensemble de patterns est noté* $\mathcal{X} = \{\mathbf{x_1}, ..., \mathbf{x_n}\}$. *Le* i^{ieme} *pattern est noté* $\mathbf{x_i} = (x_{i,1}, ..., x_{i,d})$. *En général,* $\mathcal{X}$ *peut être vu comme une matrice de dimension* $n \times d$.

5.2 Mesures de rapprochement

La majorité des algorithmes de clustering sont basés sur une notion de mesure pour effectuer le regroupement (ou la séparation) des objets. On distingue principalement deux mesures pour dégager un rapprochement entre deux objets. La plus immédiate est la notion de distance. La distance va utiliser un vecteur d'attributs de dimension constante d. Le coefficient de corrélation sera lui aussi utilisé avec cette approche de vecteur d'attributs. La deuxième mesure est celle du coefficient de similarité qui fait intervenir des notions de présences ou d'absences communes. Ces coefficients peuvent être transformé en semi-métrique.
Nous indiquons ici quelques rappels de topologie pour définir une distance.
Une distance $d_{i,j}$ dans $\mathbb{R}^p$ entre x_i et x_j est une application : $i \times j \rightarrow d_{i,j}$ qui doit vérifier :
la positivité : $d_{i,j} = d(x_i, x_j) \geq 0$,
la réflexivité : $d_{i,j} = 0 \Leftrightarrow x_i = x_j$,
l'identité : $d_{i,i} = 0$,
la symétrie : $d_{i,j} = d_{j,i}$,
l'inégalité triangulaire (pour les métriques) : $d_{i,k} + d_{k,j} \geq d_{i,j}$.

Dans certains cas, on utilise la similarité qui permet de mesurer la ressemblance entre objets et qui est plus souple que la notion de distance.
Une similarité $S_{i,j}$ dans $\mathbb{R}^p$ entre x_i et x_j est une application : $i \times j \rightarrow S_{i,j}$ qui doit vérifier :
la positivité : $S_{i,j} \geq 0$
la symétrie : $S_{i,j} = S_{j,i}$.
Plus les objets sont similaires, plus la similarité est importante.

		Objet 1	
		Présence	Absence
Objet 2	Présence	**a** = présence conjointe (1,1)	**b** = mismatch (0,1)
	Absence	**c** = mismatch (1,0)	**d** = absence conjointe (0,0)

TAB. 5.1 – Tableau de contingence : les notations.

Exemples :
La distance euclidienne est utilisée pour les attributs continus :

$$d_2(\mathbf{x_i}, \mathbf{x_j}) = \big(\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2\big)^{\frac{1}{2}} = \|\mathbf{x_i} - \mathbf{x_j}\|_2$$

qui est le cas particulier à p=2 de la métrique de Minkowski :

$$d_p(\mathbf{x_i}, \mathbf{x_j}) = \big(\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^p\big)^{\frac{1}{p}} = \|\mathbf{x_i} - \mathbf{x_j}\|_p$$

La distance de Mahalanobis :

$$d_p(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{(\mathbf{x_i} - \mathbf{x_j})' S^{-1} (\mathbf{x_i} - \mathbf{x_j})^T}$$

où $\mathbf{x_i}, \mathbf{x_j}$ sont des vecteurs colonnes et S est la covariance de la matrice des exemples. Cette distance permet d'atténuer la corrélation entre les attributs.
La distance de Hamming est aussi fortement utilisée. Il existe de nombreuses distances qui s'appliquent suivant la nature des données traitées : binaires, nominales et réelles (ou quantitatives).

5.2.1 Mesures pour les données binaires

Proximité
Les proximités construisent un tableau de contingence de taille 2 pour chaque paire d'objets. Pour simplifier les notations, nous utiliserons les conventions du tableau de contingence présenté dans la figure 5.1, où une présence est représentée par 1 ou + et une absence par 0 ou -. Les quatre cas possibles sont notés a, b, c, d.
Le tableau 5.2 présente les principales mesures de proximité utilisées dans la littérature et leurs propriétés.

Probabilités conditionnelles
Les trois mesures suivantes peuvent être interprétées comme des probabilités conditionnelles. Toutes les trois sont des mesures de similarité.

Coefficient de Similarité	**Formule**	**Exemple Si a, b, c et d =1**	**Propriétés**
Jaccard	$\frac{a}{a+b+c}$	1/3	Ne tient pas compte de l'absence conjointe
Dice	$\frac{2a}{2a+b+c}$	1/2	Ne tient pas compte de l'absence conjointe La présence conjointe est doublement pondérée
Sokal et Sneath II	$\frac{a}{a+2(b+c)}$	1/5	L'absence conjointe est ignorée Les mismatchs sont doublement comptés
Russel et Roa	$\frac{a}{d+a+b+c}$	1/4	L'absence conjointe n'est pas considérée comme une similarité
Simple Match (SM)	$\frac{d+a}{d+a+b+c}$	1/2	L'absence conjointe est considérée comme une similarité
Sokal et Sneath I (SS1)	$\frac{2(a+d)}{(2(d+a)+b+c)}$	1/3	Les présences et absences conjointes comptent doublement
Roger et Tamino (RT)	$\frac{(d+a)}{(d+a+2(b+c))}$	1/6	Les mismatchs sont doublement comptés
Sokal et Sneath III (SS3)	$\frac{a+d}{b+c}$	1	
Tanimoto	$\frac{a}{b+c}$	1/2	
Michelet	$\frac{a^2}{bc}$	1	L'absence conjointe est ignorée
Ochiaï	$\frac{a}{\sqrt{(a+b)(a+c)}}$	1/2	L'absence conjointe est ignorée
Kulczinki	$\frac{b+c}{a+d}$	1	L'absence conjointe est considérée comme une similarité

TAB. 5.2 – Mesures classiques de proximité (données binaires).

Kulczynski II (K2) mesure la probabilité conditionnelle moyenne qu'une caractéristique soit présente dans un item sachant que cette caractéristique est présente dans un autre item.
$K2(I_1, I_2) = \frac{a/(a+b)+a/(a+c)}{2}$
Sokal et Sneath IV (SS4) mesure la probabilité conditionnelle qu'une caractéristique d'un objet soit présente (ou absente) comme celle de l'autre objet.
$SS4(I_1, I_2) = \frac{a/(a+b)+a/(a+c)+d/(b+d)+d/(c+d)}{4}$
Hamann mesure la probabilité qu'une caractéristique ait le même état (présent / absent) dans les deux objets moins la probabilité qu'une caractéristique soit dans des états différents dans les deux objets. Elle est corrélée avec les mesures de proximité SM, SS1 et RT (voir tableau 5.2).
$H(I_1, I_2) = \frac{(a+d)-(b+c)}{a+b+c+d}$

Mesures de prédiction
Les quatre mesures suivantes utilisent les associations entre objets comme des prédictions de l'autre sachant l'un. Ces quatre mesures sont des similarités [Lie83].
Lambda Goodman et Kruskal
$t_1 = max(a, b) + max(c, d) + max(a, c) + max(b, d)$
$t_2 = max(a + c, b + d) + max(a + b, c + d)$
$LAMBDA(I_1, I_2) = \frac{t_1 - t_2}{a(a+b+c+d)-t_2}$
Ce coefficient mesure la réduction de l'erreur lorsqu'on utilise un objet pour en prédire un autre.
D d'Anderberg
$t_1 = max(a, b) + max(c, d) + max(a, c) + max(b, d)$
$t_2 = max(a + c, b + d) + max(a + b, c + d)$
$D(I_1, I_2) = \frac{t_1 - t_2}{2(a+b+c+d)}$
Ce coefficient mesure la réduction de la probabilité d'erreur quand un objet est utilisé pour en prédire un autre.
Q de Yule
$Q(I_1, I_2) = \frac{ad-bc}{ad+bc}$
La mesure Q de Yule est nulle si les distributions sont identiques pour les deux modalités des objets. Q est compris entre -1 (si $ad = 0$) et 1 (si $bc = 0$). Lorsque $Q = 1$ cela signifie que la présence de l'objet 1 entraîne la présence de l'objet 2 alors que si $Q = -1$ cela signifie que la présence de l'objet 1 entraîne l'absence de l'objet 2.
Y de Yule : coeffi cient de colligation
$Y(I_1, I_2) = \frac{\sqrt{ad}-\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}$
La mesure Y de Yule est une variante de la mesure Q de Yule mais elle utilise la moyenne géométrique plutôt que le nombre de paires.

5.2.2 Mesures pour les données nominales

La distance la plus utilisée pour les données de type nominale est la distance de Hamming. Soit deux objets a et b décrits par n variables nominales, la distance de Hamming peut être définie de la manière suivante :

$$d_H(a, b) = \sum_{i=1..n} d(a_i, b_i) \text{ avec } d(a_i, b_i) = \begin{cases} 1 \text{ si } a_i \neq b_i \\ 0 \text{ sinon} \end{cases}$$

Les mesures suivantes peuvent être aussi utilisées pour des variables de type nominal :

– Simple Match,
– Coefficient Kappa pour les variables nominales,
– Distance de Manhattan,
– Distance Euclidienne au carré.

5.2.3 Mesures pour les données réelles

Les mesures pour les données continues ou réelles sont nombreuses (voir tableau 5.3) et elles peuvent être classées en fonction de ce qu'elles évaluent : une dissimilarité ou une similarité.

Similarité	**Dissimilarité**
Coefficient de corrélation de Pearson	
	Distance Euclidienne
	Distance du χ^2
	Distance de Manhattan
Coefficient de rand de corrélation de Spearman	Dissimilarité de Spearman
Coefficient de rand de corrélation de Kendall	Dissimilarité de Kendall

TAB. 5.3 – Mesures de similarité et dissimilarité pour les données réelles.

La présentation de ces différentes mesures montre la diversité des mesures existantes et l'importance du choix de la distance ou de la similarité car le clustering est basé sur une notion de distance. Les données que l'on doit traiter sont souvent composées d'attributs de différents types et il est important de bien choisir la distance pour ne pas trop influencer le clustering. Ces distances peuvent aussi dépendre du problème.

5.3 Algorithmes de clustering

Il existe différents algorithmes de clustering. Nous utiliserons la taxinomie proposée par A.K. Jain, M.N. Murty et P.J. Flynn [JMF99a].

Les propriétés utilisées de cette taxinomie sont :
– Agglomérante ou divisante : cet aspect reflète la structure algorithmique et les opérations effectuées. Une approche agglomérante commence avec chaque exemple dans un cluster différent et concatène les clusters jusqu'à ce qu'un critère d'arrêt soit atteint, alors qu'une approche divisante commence avec tous les exemples dans le même cluster et subdivise ce cluster.
– Monotétique ou polytétique : cet aspect se réfère à l'utilisation séquentielle ou simultanée des attributs pour le clustering. La plupart des algorithmes sont polytétiques, c'est à dire que

tous les attributs entrent dans le calcul de la distance et les décisions sont prises par rapport à cette distance.

– Dur ou flou (hard vs fuzzy) : un algorithme de clustering dur alloue chaque exemple à un seul cluster durant les opérations qu'il effectue et pour les résultats qu'il donne en sortie, alors qu'une méthode de clustering floue donne un degré d'appartenance d'un exemple à certains clusters.
– Déterministe ou stochastique : cette notion est plus parlante pour les approches par partitionnement qui optimisent une fonction dite de "squared error". Cette optimisation peut être accomplie de manière traditionnelle ou à travers une recherche aléatoire dans un espace de recherche constitué de tous les labels de classe possibles.
– Incrémental ou non incrémental : cette notion apparaît quand l'ensemble des exemples à partitionner est grand et que les contraintes en terme de temps d'exécution ou d'espace mémoire affectent l'architecture de l'algorithme.

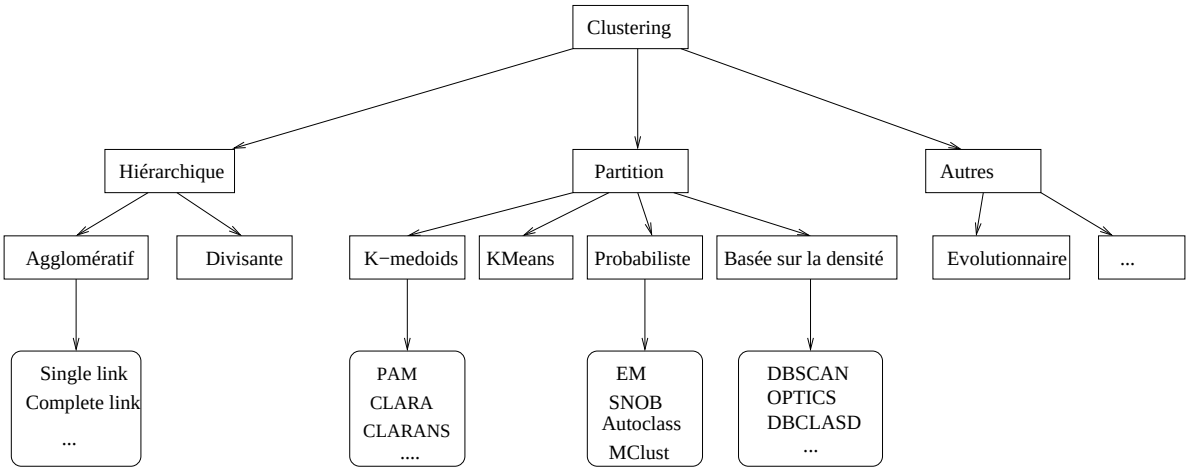


FIG. 5.1 – Taxinomie des algorithmes de clustering.

Nous présenterons dans cette partie uniquement les algorithmes de clustering qui seront utilisés dans la suite de ce mémoire.

5.3.1 Algorithmes hiérarchiques

L'agglomération [Gar98] ou classification hiérarchique regroupe deux à deux les éléments les plus proches de sorte à former de nouveaux éléments que l'on regroupe à leur tour. Pour déterminer les éléments les plus proches, on construit la matrice des distances entre les n éléments et l'on extrait récursivement les plus proches. Chaque groupe est ensuite représenté par un seul élément (par exemple son centre).
La plupart des algorithmes hiérarchiques sont des variantes des algorithmes "single link", "complete link" et "minimum variance".
Les algorithmes "single-link" et "complete-link" sont les plus populaires mais ils diffèrent l'un de l'autre par la manière dont est caractérisée la similarité entre une paire de clusters. Dans l'algo-

rithme single-link, la distance entre deux clusters est représentée par la distance minimum entre toutes les paires de données des deux clusters C_i et C_j (la paire étant composée d'un élément de chaque cluster), on parle alors de saut minimum $D_{min}(C_i, C_j) = min_{x_i \in C_i, x_j \in C_j} d(x_i, x_j)$. Dans l'algorithme "complete-link", la distance entre deux clusters est le maximum de la distance entre tous les exemples des deux clusters, on parle alors de saut maximum $D_{max}(C_i, C_j) = max_{x_i \in C_i, x_j \in C_j} d(x_i, x_j)$. Une illustration des calculs de distance est donné dans la figure 5.2.
L'algorithme "average link" propose, quant à lui, de calculer la similarité en prenant la valeur moyenne des distances entre tous les couples d'objets des deux clusters. L'algorithme "average link" se révèle coûteux pour des bases de données volumineuses.

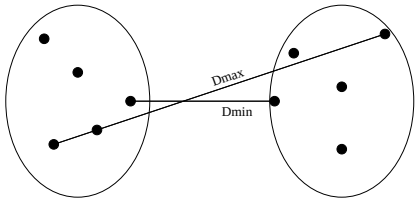


FIG. 5.2 – Exemple de calculs de D_{max} et D_{min}.

La figure 5.3 représente l'application de l'algorithme afin de classer les éléments {20, 30, 46, 52, 57, 60}.

Il a été observé que l'algorithme "complete-link" produit une hiérarchie plus intéressante dans de nombreux cas que l'algorithme "single-link".

Dist	20	30	46	52	57	60
20	0	10	26	32	37	40
30	10	0	16	22	27	30
46	26	16	0	6	11	14
52	32	22	6	0	5	8
57	37	27	11	5	0	3
60	40	30	14	8	3	0

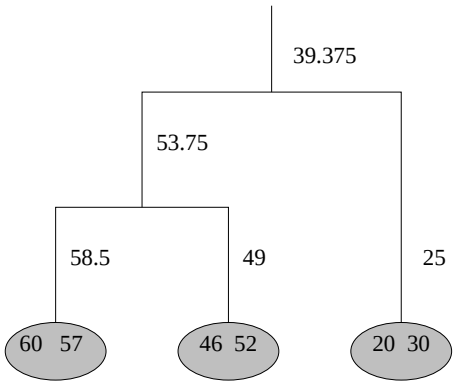


FIG. 5.3 – Exemple d'agglomération hiérarchique.

5.3.2 Algorithmes à partitionnement

Ces algorithmes partitionnent les objets en différents groupes. Il existe différents algorithmes à partitionnement, nous détaillerons dans cette section les plus classiques : Kmeans et Kmedioïd.

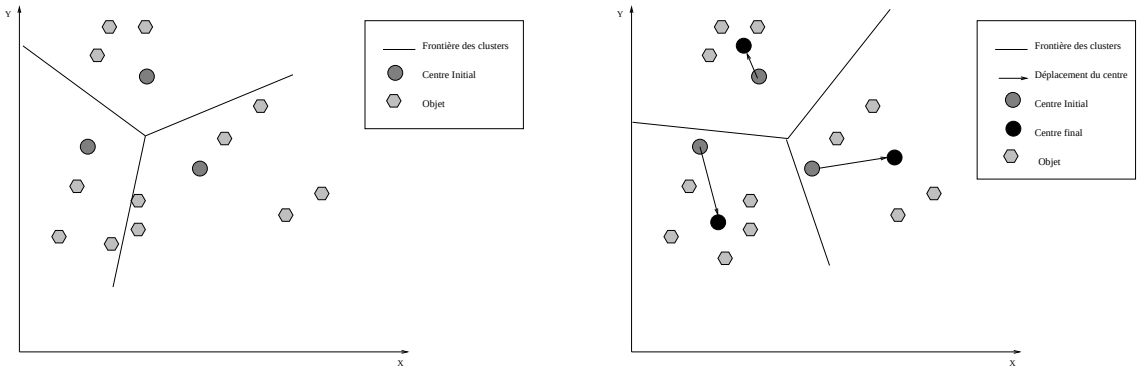


FIG. 5.4 – Illustration de Kmeans dans le plan : déplacement des centres et des frontières des clusters.

Kmeans
L'algorithme Kmeans utilise des principes heuristiques minimisant l'erreur quadratique. Il requiert une partition initiale et un nombre de clusters K à construire. Cette méthode, encore appelée méthode des centres mobiles [LMP98], permet de classer des objets en N groupes en les agrégeant autour de centres. L'algorithme choisit d'abord K objets initiaux qui seront les centres des K groupes, ensuite il place chaque objet dans le groupe de centre le plus proche, puis recalcule le centre de chaque groupe et ainsi de suite jusqu'à ce que les objets ne changent plus de groupe. La figure 5.4 donne une illustration de Kmeans dans le plan.
L'algorithme général est présenté dans [GT00].
Il est prouvé mathématiquement que l'algorithme Kmeans converge vers un minimum local qui dépend des centres initiaux [BB95, LMP98]. L'algorithme converge car la variance à l'intérieur d'une classe ne peut que décroître ou rester stationnaire entre deux étapes successives [LMP98].
Le problème de trouver un minimum global est NP-complet.

Kmedioïd
L'algorithme des Kmedioïd travaille uniquement avec les objets à partitionner. Il nécessite dans sa version basique de fixer le nombre K de clusters que l'on veut créer. Il travaille itérativement en déplaçant les objets jusqu'à ce que la solution obtenue soit stable. Cet algorithme minimise la fonction de compacité TD définie par $TD = \sum_{i=1}^{k} \sum_{p \in C} d(p, C_c)$ avec C_c le représentant du cluster C. Cet algorithme est un peu moins sensible au bruit que Kmeans.

5.3.3 Comparaison des algorithmes en terme de complexité

En général, les algorithmes hiérarchiques sont beaucoup plus coûteux en temps et en espace que les algorithmes par partitionnement.
Soit n les nombre d'exemples, k le nombre de clusters et l le nombre d'itérations prises par l'algorithme pour obtenir une convergence, le tableau 5.4 récapitule les complexités en temps et en espace des différents algorithmes présentés.

Algorithme de clustering	Complexité en temps	Complexité en espace
Kmeans	$O(nkl)$	$O(k)$
Kmedioïd	$O(nkl)$	$O(k)$
Single-link	$O(n^2 \log n)$	$O(n^2)$
Complete-link	$O(n^2 \log n)$	$O(n^2)$
Average Link	$O(n^2)$	$O(n^2)$

TAB. 5.4 – Comparaison de la complexité en temps et en espace de différents algorithmes de clustering.

5.4 Les critères de clustering

L'évaluation de la qualité d'un clustering n'est pas triviale. Tout comme il existe de nombreux algorithmes de clustering se basant sur des techniques d'optimisation, il existe aussi différents critères de clustering utilisés dans la littérature [ZK01, Gue03]. Nous ne nous intéresserons ici qu'aux critères dits internes c'est à dire basés sur les données et les similarités alors que les critères externes sont basés sur des informations externes comme le label de classes ou l'avis d'un expert. Lorsque l'on utilise les critères internes, le clustering est alors considéré comme un problème d'optimisation et on peut déterminer la performance d'une clusterisation. La présentation des critères se fera selon qu'ils travaillent avec un nombre de classes fixé ou non.

5.4.1 Nombre de clusters fixé

Les critères de qualité de clustering, lorsque le nombre de clusters est fixé, sont relativement nombreux. Nous ne citerons que le plus connu car, dans la suite de notre travail, nous ne nous intéressons qu'au clustering avec un nombre de clusters inconnu.

Mean Squarred Error

La fonction de coût la plus populaire provient de la mesure de l'éparpillement entre les points de chaque cluster. Le coût est mesuré comme la moyenne de la distance entre les points et le centre du cluster auquel ils appartiennent. Cette fonction est minimisée dans l'algorithme $k$means.

$$MSE = \frac{1}{kT} \sum_{i=1} N \sum_{j=1}^{T_i} d(x_j^{(i)}, C_i)$$

avec T_i : nombre d'objets dans le cluster i, T : nombre total d'objets, $x_j^{(i)}$: le j$^{\text{ème}}$ exemple du i$^{\text{ème}}$ cluster, k : nombre de clusters.

5.4.2 Nombre de clusters non fixé

Déterminer le nombre optimal de clusters pour un jeu de données est l'un des aspects les plus difficiles du processus de clustering. La plupart des méthodes d'optimisation présentées jusqu'ici supposent que l'utilisateur spécifie le nombre de clusters alors que les méthodes hiérarchiques produisent une série de clustering de 1 à n clusters sans spécifier le nombre de clusters le plus

approprié.
De nombreuses procédures ont été suggérées pour déterminer le nombre optimal de clusters incluant quelques critères pouvant être utilisés comme fonction objective pour des méthodes d'optimisation [DB79, CH74, MC85].
Nous nous servirons de ces indices pour comparer notre méthode à celle développée par Bandyopadhyay et Maulik [BM01] sur des jeux de données numériques.
Pour la présentation des différents indices, les notations sont les suivantes : c représente le nombre de clusters, n le nombre d'attributs, χ_i le $i^{\text{ème}}$ cluster, z_i le centre du $i^{\text{ème}}$ cluster et $d(x,y)$ la distance entre deux objets.
L'indice *DB* [DB79] est une fonction basée sur la minimisation du rapport des dispersions intra-clusters S_i et de la séparation inter-clusters R_i, introduite par Davis et Bouldin. L'éparpillement du $i^{\text{ème}}$ cluster est calculé comme suit :

$$S_{i,q} = \left(\frac{1}{|\chi_i|} \sum_{x \in \chi_i} ||x - z_i||_2^q \right)^{\frac{1}{q}}$$

$$R_{i,qt} = \max_{j,\, j \neq i} \left\{ \frac{S_{i,q} + S_{j,q}}{d_{i,j,t}} \right\} \text{ avec } d_{i,j,t} = ||z_i - z_j||_t$$

$$DB = \frac{1}{c} \sum_{i=1}^{c} R_{i,qt}$$

avec q et t fixés par l'utilisateur pour pouvoir pondérer la fonction.
Donc, le ratio est petit si les clusters sont compacts et éloignés les uns des autres. Par conséquence, l'indice de Davies-Bouldin aura une petite valeur quand le clustering sera de bonne qualité.

L'indice de *Dunn* v_D [Dun73] est basé sur l'identification d'ensemble de clusters compacts et bien séparés. Soient S et T deux sous-ensembles non vides de $\mathbb{R}^n$. Le diamètre Δ de S et la distance δ sont alors :

$$\Delta(S) = \underbrace{\max}_{x,\, y \in S} \{d(x,y)\}$$

$$\delta(S,T) = \underbrace{\min}_{x \in S, y \in T} \{d(x,y)\}$$

L'indice de Dunn est alors :

$$v_D = \underbrace{\min}_{1 \leq i \leq c} \left\{ \underbrace{\min}_{1 \leq j \leq c,\, j \neq i} \left\{ \frac{\delta(\chi_i, \chi_j)}{\underbrace{\max}_{1 \leq k \leq c} (\Delta(\chi_k)} \right\} \right\}$$

L'objectif principal de cet indice est de maximiser la distance inter-cluster et de minimiser la distance intra-cluster. L'objectif est donc de maximiser l'indice de Dunn.

L'indice de *Dunn généralisé* v_{GD} [Bez98] a été introduit par Bezdek après qu'il ait démontré que l'indice de *Dunn* est sensible aux changements dans la structure des clusters. Les changements introduits interviennent dans le calcul de Δ et δ.

Nous ne nous intéresserons qu'à trois mesures, Δ_3, δ_3, δ_5 jugées les plus intéressantes par Bezdek pour vérifier la qualité des clusters trouvés.

$$\Delta_3 = 2\left(\frac{\sum_{x\in S} d(x, z_S)}{|S|}\right)$$

$$\delta_3(S,T) = \frac{1}{|S||T|}\sum_{x\in S, y\in T} d(x,y)$$

$$\delta_5(S,T) = \frac{1}{|S|+|T|}\left(\sum_{x\in S} d(x, z_T) + \sum_{y\in T} d(y, z_S)\right)$$

Cela nous donne deux indices possibles v_{33} et v_{53}.
L'indice $\mathcal{I}$ [BM01] est une combinaison de trois facteurs introduit récemment par Bandyopadhyayis.

$$I(c) = \left(\frac{1}{c} \times \frac{E_1}{E_c} \times D_c\right)^p$$

avec

$$E_c = \sum_{k=1}^{c}\sum_{j=1}^{n} u_{kj}||x_j - z_k||$$

$$D_c = \max_{i,j=1}^{c} ||z_i - z_j||$$

$U(X) = [u_{kj}]_{c\times n}$ représente la matrice des partitions de données.
Le premier facteur essaie de diminuer le nombre de clusters c alors que les deux autres tentent d'augmenter c. Cet indice encourage la formation de clusters compacts et bien séparés.

Milligan et Cooper comparent 30 critères pour déterminer le nombre de clusters [MC85]. Le critère de Calinski et Harabasz (CH) produit, sur les jeux de tests qu'ils ont utilisés, les meilleurs résultats [CH74]. Soient c clusters, n attributs, soit χ_j le $j^{\text{ème}}$ cluster avec j=1..c,
Soient m le vecteur de moyenne sur les données $m = \begin{pmatrix} m_1 & m_2 & \cdots & m_n \end{pmatrix}$
et m_j le vecteur des moyennes pour le $j^{\text{ème}}$ cluster $m_j = \begin{pmatrix} m_{j_1} & m_{j_2} & \cdots & m_{j_n} \end{pmatrix}$.
X_i est un élément du cluster χ_j et $(X_i - m_j)^t$ est la transposée du vecteur $(X_i - m_j)$ i.e.

$$(X_i - m_j)^t = \begin{pmatrix} X_1 - m_{j_1} \\ X_2 - m_{j_2} \\ \cdots \\ X_n - m_{j_n} \end{pmatrix}.$$

La matrice d'éparpillement pour le $j^{\text{ème}}$ cluster est alors : $P_j = \sum_{X_i\in\chi_j}(X_i - m_j)(X_i - m_j)^t$.
La matrice d'éparpillement intra-cluster pour c clusters :

$$P_{Within} = \sum_{j=1}^{c} P_j \tag{5.1}$$

et la matrice d'éparpillement inter-clusters :

$$P_{Between} = \sum_{j=1}^{c} n_j(m_j - m)(m_j - m)^t \tag{5.2}$$

Le critère de Calinski et Harabasz est alors :

$$CH = \frac{(n-c) \times trace(P_{Between})}{(c-1) \times trace(P_{Within})} \tag{5.3}$$

Il existe plusieurs fonctions dont l'objectif est tout simplement de minimiser la distance entre le centre du cluster et les objets le composant.

$$f(P,C) = \sum_{i=1}^{K} d(x_i, c_{p_i})^2$$

Cette fonction est identique au critère utilisé dans la méthode de Ward [EHW86].

Dans [Bar89, BH90, BH92], les auteurs proposent d'adapter la fonction relative à la distance intra-cluster pour un nombre de clusters inconnu. Ils proposent de minimiser le critère de Barker [Bar89] sur une partition P de k clusters :

$$B(P) = \sum_{j=1}^{k} \sum_{o_x \in j} \sum_{\substack{o_y \in j \\ x \neq y}} (\sigma(o_x, o_y) - v)$$

où v est un seuil de distance fixé par les utilisateurs, la distance σ doit vérifier que quelques soient deux objets o_x et o_y : $\sigma(o_x, o_x) = 0$, et que $\sigma(o_x, o_y) \geq 0$.
Comme $\sigma(o_x, o_y) - v$ est positif pour $\sigma(o_x, o_y) > v$, l'algorithme tend à produire des clusters qui ont un diamètre inférieur à v. Il faut donc supposer que l'on choisit un seuil v tel que deux objets dont la distance est supérieure à v n'aient aucune raison d'être ensemble. Cette fonction, par l'utilisation du seuil v, reprend les mêmes contraintes que dans ISODATA [BH65]. ISODATA est en effet un algorithme basé sur l'approche par centre mobile qui introduit, au cours des itérations successives, de nouveaux paramètres afin de modifier le nombre de classes : si la distance intra-groupes est trop grande, on peut diviser la classe en deux et recalculer la position des centres de gravité des deux noyaux ainsi obtenus. Si la distance inter-groupes est trop petite, on peut fusionner les classes et calculer, là aussi, le nouveau centre de gravité du nouveau groupement.
L'algorithme propose en effet par l'utilisation de la fonction de Baker :

- la pénalisation des petits clusters car v sera soustrait moins de fois,
- la pénalisation des grands clusters car $\sigma(o_x, o_y)$ sera ajouté plus de fois.

La matrice de similarité est pré-calculée et la complexité de l'algorithme est relativement faible puisque $\sigma(o_x, o_y) - v$ est connu quelques soient x et y.

5.5 Complexité du problème de clustering

Le nombre de façons de classer n objets dans k groupes est donné par Liu [Liu68] :

$$N(n,k) = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \begin{pmatrix} k \\ i \end{pmatrix} (k-i)^n$$

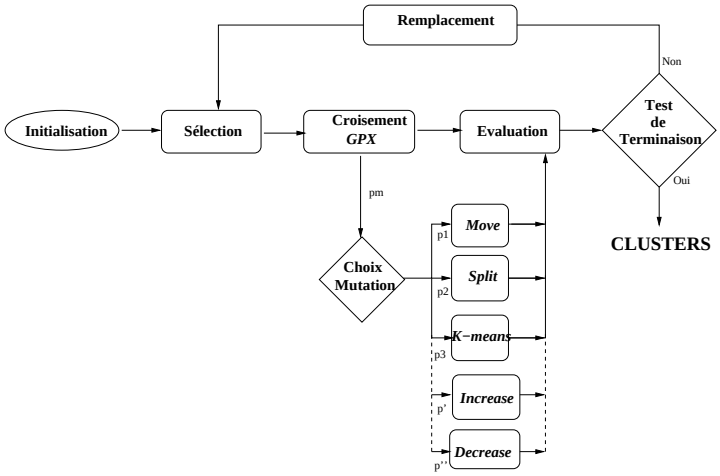


FIG. 5.5 – Les différentes étapes de l'algorithme génétique CHyGA.

Par exemple, si nous prenons n=25 et k=5, il y a 2 436 684 974 110 751 façons de répartir les 25 objets dans 5 groupes [And73]. Si le nombre de clusters est inconnu (ce qui est souvent le cas dans les applications issues de cas réels), les objets peuvent être répartis de $\sum_{i=1}^{n} N(i,k)$ façons. Soit pour l'exemple précédent 4×10^{18}. Il est donc évident qu'un parcours exhaustif des solutions est impossible.
Les méthodes traditionnelles ne travaillent que sur un petit sous-ensemble de l'espace de recherche (le sous-ensemble étant défini par le nombre de clusters, le critère de clustering et la méthode de clustering). Ces méthodes traditionnelles obtiennent donc, en général, des optima locaux et rarement globaux.
Les métaheuristiques, ayant déjà fait leurs preuves pour la résolution de problèmes combinatoires de grandes tailles, peuvent donc sembler intéressantes pour se dégager de ces optima locaux et trouver de façon plus fréquente les optima globaux.

5.6 CHyGA : un algorithme de clustering

CHyGA est un algorithme génétique que nous proposons pour réaliser un clustering aussi bien sur des données réelles que sur des données nominales. Cet algorithme travaille à partir d'une représentation double, adaptée et optimisée pour le problème, que nous présenterons. Il utilise l'algorithme Kmeans dans le cadre d'une hybridation. Nous présenterons tout d'abord son fonctionnement général puis les adaptations qui sont faites pour le mécanisme d'hybridation avec Kmeans pour traiter les différents types de données.
La figure 5.5 récapitule les différentes étapes de l'algorithme génétique mis en place pour traiter le problème du clustering.

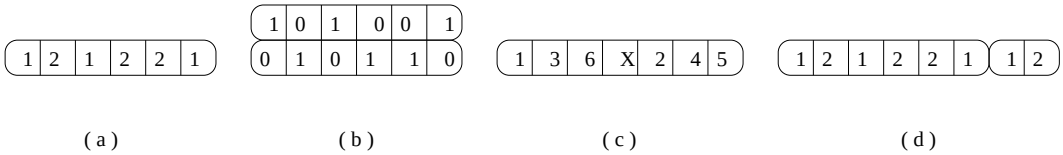


FIG. 5.6 – Différentes représentations utilisées dans la littérature pour coder le clustering $\{\{X_1, X_3, X_6\}, \{X_2, X_4, X_5\}\}$. (a) group number, (b) matrice, (c) permutation avec représentation du séparateur, (d) GGA.

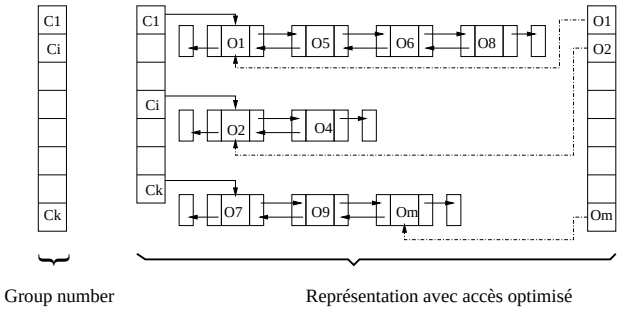


FIG. 5.7 – Le codage d'une solution : représentation hybride.

5.6.1 Représentation d'une solution

Pour les problèmes de clustering ou de partition, il existe plusieurs représentations possibles [Mic96] :
- la représentation "group number" qui indique pour chaque objet le groupe auquel il appartient [JB91a],
- la représentation par matrice où chaque ligne représente un cluster et chaque colonne est associée à un objet [BBHB94]. Un 1 représente la présence de cet objet dans le cluster et un zéro son absence,
- la permutation avec représentation du séparateur où un caractère est utilisé pour définir les frontières des clusters [JB91b],
- la représentation GGA (Grouping Genetic Algorithm) qui reprend la représentation standard "group number" et rajoute une partie qui récapitule les groupes en présence [Fal98].

La figure 5.6 donne les codages du clustering $\{\{X_1, X_3, X_6\}, \{X_2, X_4, X_5\}\}$ pour les différentes représentations présentées. Les clusters sont numérotés un et deux et les six objets sont identifiés par leur numéro. Toutes ces représentations ont leurs avantages et leurs inconvénients.
Dans notre cas, nous avons choisi une double représentation hybride (voir figure 5.7) qui regroupe la représentation "group number" et la représentation du contenu des clusters. Cette représentation permet de trouver rapidement l'affection d'un objet et la composition des clusters.
Dans la structure "group number", pour chaque objet nous indiquons le cluster auquel il appartient. Dans la structure complexe, pour chaque cluster une liste doublement chaînée donne sa composition ce qui permet des modifications rapides de la composition. De plus, un tableau permet de retrouver directement chaque objet (et de la bouger d'un cluster à l'autre en temps constant).

5.6.2 Fonction d'évaluation

Dans les sections 5.4 et 5.4.2 nous avons présenté différents critères permettant d'évaluer la qualité d'un partitionnement. Le critère de Calinski et Harabasz (voir paragraphe 5.4.2) est celui donnant les meilleurs résultats dans la littérature [CH74]. Il mesure la bonne distribution des objets dans les clusters.

Nous rappelons le critère de Calinski et Harabasz :

$$CH = \frac{(n-k) \times trace(P_B)}{(k-1) \times trace(P_W)} \tag{5.4}$$

5.6.3 Opérateurs

Le choix des opérateurs de croisement et de mutation utilisés a été guidé par le codage utilisé. Notre algorithme génétique comporte deux types d'opérateurs : opérateurs de croisement et opérateurs de mutation.

5.6.3.1 Croisement

Nous utilisons le croisement GPX [GH99] (Greedy Partition Crossover) qui est un croisement adapté aux partitions (voir figure 5.8). Soit deux parents p_1 et p_2, l'algorithme construit un enfant o comme décrit dans l'algorithme 8 où k est le nombre maximal de clusters (certains clusters peuvent être vides).
Notre représentation est bien adaptée pour cet opérateur car les éléments d'un même cluster sont regroupés.

Algorithme 8 GPX Crossover

Données : 2 parents $p_1 = \{C_1^1, ..., C_k^1\}$ and $p_2 = \{C_1^2, ..., C_k^2\}$
Résultat : un enfant $o = \{C_1, ..., C_k\}$
for l(1 $\leq$ l $\leq$ k) **do**
  if l est impair **then**
    A :=1
  else
    A :=2
  end if
  choisir i tel que C_i^A ait la cardinalité maximale
  $C_l := C_i^A$
  enlever les objets de C_l dans p_1 et p_2
end for
Affecter aléatoirement les objets appartenant à aucun cluster

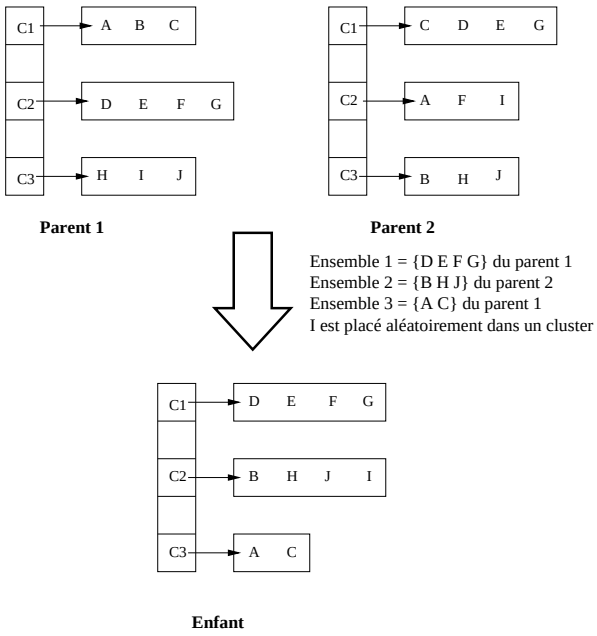


FIG. 5.8 – L'opérateur de croisement : un exemple.

5.6.3.2 Mutations simples

Pour les problèmes de clustering, comme celui que nous avons voulu traiter, les opérateurs de mutation doivent être capables d'introduire de nouveaux clusters et d'enlever des clusters existants. Dans [Fal94], Falkenauer remarque que la mutation d'un seul gène n'est pas suffisante car cela n'augmentera pas significativement la fonction d'évaluation et la mutation sera vite perdue dans la population. Notre algorithme génétique utilise trois opérateurs de mutations différents quelques soient la nature des objets.
Les trois différents opérateurs de mutation que nous proposons sont "split" [Col98], "move" [Col98] et une itération de Kmeans (décrite dans le paragraphe 5.6.5).
L'opérateur "split" choisit un groupe d'un cluster particulier et déplace les objets de ce groupe vers un nouveau cluster avec une probabilité fixée. L'opérateur "move" échange des objets entre deux clusters existants (voir figure 5.9).

5.6.3.3 Mutations élaborées

Dans le cadre de ce travail, nous avons mis en place des mutations répondant à notre problème. Nous avons, tout d'abord, implémenté un opérateur permettant d'augmenter le nombre de clusters, "increase". Cette mutation choisit le cluster ayant la plus grande variance interne pour le diviser en deux clusters. Les objets sont alors redistribués entre deux nouveaux clusters. Pour cela, on construit deux nouveaux centres : le premier séparant en deux le maximum de chaque attribut et la valeur moyenne et le second séparant en deux le minimum et la moyenne, puis on affecte les

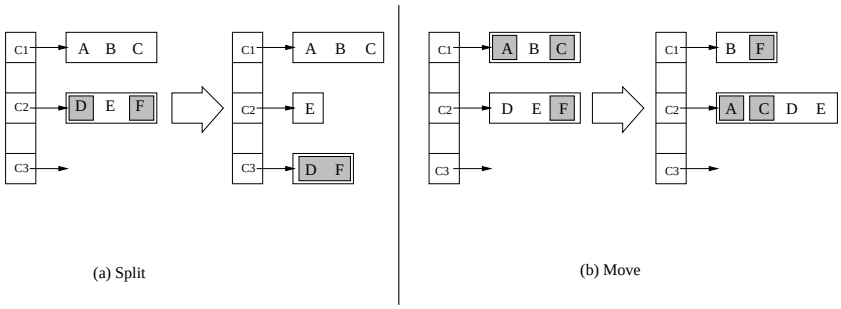


FIG. 5.9 – Les opérateurs de mutation : Split et Move.

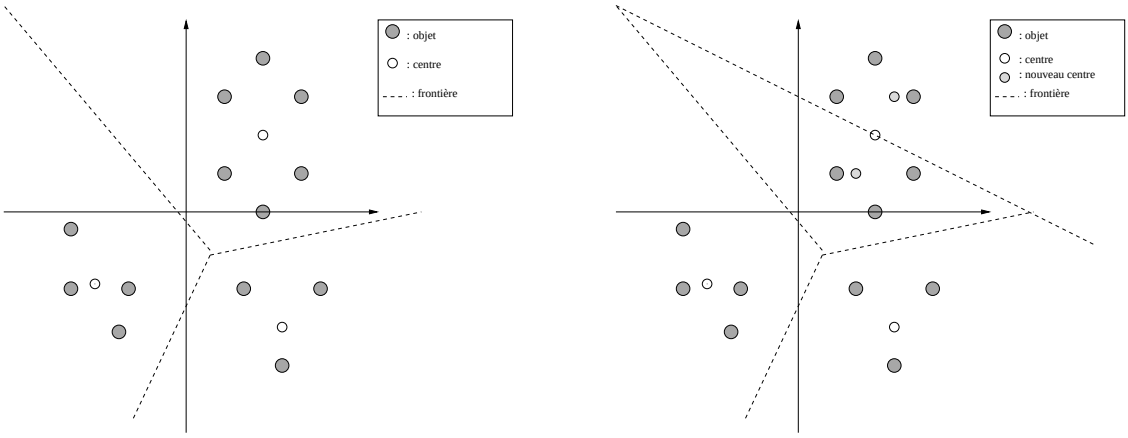


FIG. 5.10 – Illustration de l'opérateur "increase" dans le plan.

objets en fonction de leur écart au centre (l'objet va dans le centre le plus proche). La figure 5.10 illustre le fonctionnement de l'opérateur "increase".
Notre second opérateur de mutation, "decrease" reprend l'idée du clustering hiérarchique et a pour objectif de regrouper deux clusters pour n'en former qu'un. L'opérateur détecte les deux centres les plus proches et regroupe les deux clusters associés en un seul.

5.6.3.4 Mécanisme de diversification

L'un des défauts que l'on peut remarquer lors de l'évolution de l'algorithme génétique est une stagnation de la recherche. Pour remédier à cela, nous avons mis en œuvre, comme pour les applications du chapitre 4.2.2.3, un mécanisme de diversification appelé migration de diversité stochastique ou "Random Immigrant". Nous pouvons observer sur la figure 5.11 les apparitions de faille dans la courbe d'évolution de la moyenne des notes lors de la mise en œuvre du mécanisme.

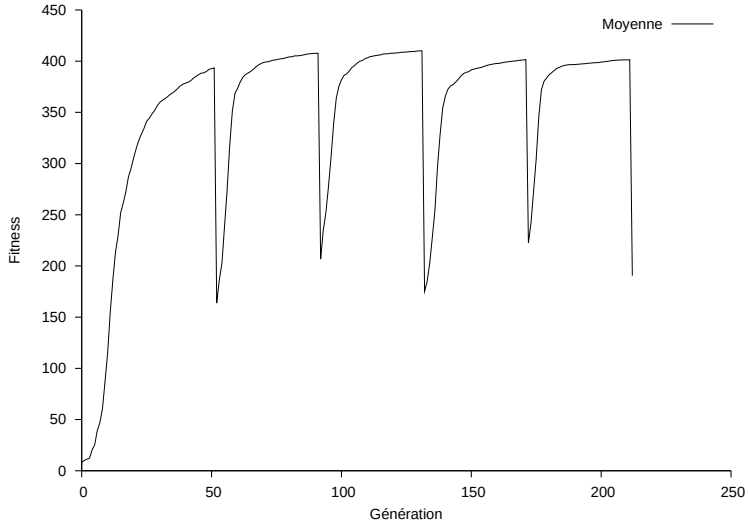


FIG. 5.11 – Évolution de la population sur le jeu de données Iris : mise en œuvre du random Immigrant.

5.6.4 Données manquantes

De nombreux jeux de données comportent des données manquantes. La stratégie employée dans Chyga pour les traiter est commune à d'autres travaux [JMF99b]. Nous ne comptons pas l'attribut manquant dans le calcul de la distance qui représente donc une moyenne des distances de l'ensemble des attributs connus.

5.6.5 Mise en place de l'hybridation

Nous avons vu précédemment que l'algorithme Kmeans était une recherche locale. Dans le cadre de ce travail, nous avons décidé d'utiliser le principe de cet algorithme pour réaliser une mutation en effectuant une itération de l'algorithme (voir schéma de l'algorithme figure 5.5). Nous avons voulu nous limiter à une itération car comme nous désirions que notre algorithme soit adaptable à de grandes bases de données, et nous n'avons donc pas voulu que Kmeans se déroule entièrement. La complexité de Kmeans dépend, en effet, de la taille des données et du nombre de clusters.
L'algorithme ayant été proposé initialement pour des données numériques, la mise en place de l'hybridation avec celui-ci a fait l'objet d'une réflexion sur la représentation d'un centre lorsque les données sont nominales et la proposition d'une distance associée.

5.6.5.1 Définition d'un centre dans le cas nominal

Le centre d'un cluster est facile à définir lorsque l'on travaille sur des données réelles car la moyenne a une significativité. Dans quelques travaux, des auteurs proposent des méthodes pour choisir le centre d'un cluster dans le cas des données nominales. Huang propose de se rapprocher

de la simplicité du calcul de la moyenne en proposant d'utiliser le mode de la variable nominale et donc de choisir comme représentant de cette variable celui qui a la plus grande fréquence [Hua98].

Définition 17 *Soit X un ensemble d'objets décrits par des attributs* $A_1, ..., A_m$.
Un mode de X est le vecteur $Q \in \Omega$, $Q = [q_1 q_2 ... q_m]$ *qui minimise* $D(Q, X) = \sum_{i=1}^{n} d(X_i, Q)$.
Q n'est pas forcément un élément de X.

L'inconvénient du mode est double : Q n'est pas forcément unique, comment alors en choisir un ? De plus, si nous prenons une répartition de valeurs pour un attribut comme celle-ci : "$Y, ?, Y, N, Y, N, ?$" le mode choisira la valeur Y ayant une fréquence de 3 ce qui n'a pas une grande significativité.

Nous avons donc décidé de ne pas utiliser ce type d'élection de centre. Nous proposons de considérer que la valeur d'un attribut doit être choisie par un vote majoritaire et lorsqu'il n'y a pas de candidat satisfaisant cette contrainte nous disons que le centre n'est que partiellement défini. Nous utiliserons alors la notation *.

Définition 18 *Soit X un ensemble d'objets décrits par des attributs* $A_1, ..., A_m$.
Le centre de X est le vecteur $Q = [q_1 q_2 ... q_m]$ *avec* $q_i \in \Omega \cup [*]$ *qui minimise* $D(Q, X) = \sum_{i=1}^{n} d(X_i, Q)$. *Q n'est pas forcément un élément de X. Q est unique.*

Le centre, ainsi défini, sera appelé centre partiel et représente un hyperplan au lieu d'un point de l'espace $\mathbb{R}^m$ comme par exemple dans le cas d'un mode. Dans l'exemple d'un centre Q_i défini par [1, *, *] en dimension 3, le centre représente le plan $\mathcal{P}$ (voir figure 5.12). La dimension de l'hyperplan ainsi défini dans $\mathbb{R}^{\text{Nb attributs}}$ est donc *nombre total d'attributs moins nombre d'attributs déterminés du centre*.

5.6.5.2 Définition d'une distance

La distance entre objets est basée sur une distance de Hamming.
Étant donné que le centre est partiellement défini, comment évaluer l'écart d'un objet au centre ?
Posons $d_v(O, C)$ la distance d'un centre à un objet. Nous définissons $d_v = \sum_i d_\sigma(O_i, C_i)$ défini par :

$$\forall x \text{ et } y, \quad d_\sigma(x, y) = \begin{cases} 0 \text{ si x=y} \\ 1/2 \text{ si x ou y vaut * (avec x}\neq\text{y)} \\ 1 \text{ si x} \neq \text{ y} \end{cases}$$

Nous montrons ci-dessous que d_v proposée vérifie les propriétés d'une distance.

Preuve 1 *Positivité*
Par construction, d_σ *est toujours positive ou nulle donc*
$d_v(O, C) = \sum_i d_\sigma(O_i, C_i) \geq 0$.

Preuve 2 *Symétrie*
$d_v(O, C) = \sum_i d_\sigma(O_i, C_i)$ *donc* d_v *est symétrique si* d_σ *l'est.*

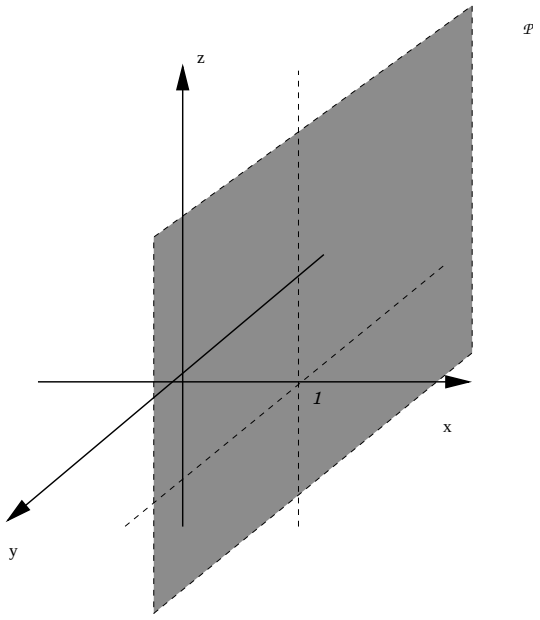


FIG. 5.12 – L'hyperplan $\mathcal{P}$ défini par le centre [1 * *] dans $\mathbb{R}^3$.

Considérons les différentes possibilités :
Si $d_\sigma(O_i, C_i) = 0 \Longrightarrow O_i = C_i$ *ou encore,* $C_i = O_i$ *donc* $d_\sigma(C_i, O_i) = 0$.
Si $d_\sigma(O_i, C_i) = \frac{1}{2} \Longrightarrow C_i = *$ *puisque* O_i *est toujours défini, donc* $d_\sigma(C_i, O_i) = \frac{1}{2}$.
Si $d_\sigma(O_i, C_i) = 1 \Longrightarrow O_i \neq C_i$ *donc* $C_i \neq O_i$ *donc* $d_\sigma(C_i, O_i) = 1$.
d_σ *est symétrique donc* d_v *est symétrique.*

Preuve 3 *Inégalité triangulaire*
Soient O_1, O_2, O_3 *trois objets pouvant être des centre. Montrons que* $d_v(O_1, O_2) \leq d_v(O_1, O_3) + d_v(O_3, O_2)$. *Il faut donc que* $\sum_i d_\sigma(O_{1_i}, O_{2_i}) \leq \sum_i d_\sigma(O_{1_i}, O_{3_i}) + \sum_i d_\sigma(O_{3_i}, O_{2_i})$.
Soient $A = \sum_i d_\sigma(O_{1_i}, O_{2_i})$ *et* $A_i = d_\sigma(O_{1_i}, O_{2_i})$.
Soient $B = \sum_i d_\sigma(O_{1_i}, O_{3_i}) + \sum_i d_\sigma(O_{3_i}, O_{2_i})$ *et* $B_i = d_\sigma(O_{1_i}, O_{3_i}) + d_\sigma(O_{3_i}, O_{2_i})$.
On calcule attribut par attribut :
Si $O_{1_i} = O_{2_i}$ *alors* $A_i = 0$ *:*
- *Si* $O_{1_i} = O_{3_i}$ *alors* $O_{2_i} = O_{3_i}$ *alors* $B_i = 0$
- *Si* $O_{1_i} \neq O_{3_i}$ *alors* $B_i += 1/2$ *ou* 1 *et alors* $O_{3_i} \neq O_{2_i}$ *alors* $B_i += 1/2$ *ou* 1

$\Longrightarrow B_i = 0, 1$ *ou* 2. *Dans ce cas* $A_i \leq B_i$.

Si $O_{1_i} \neq O_{2_i}$ *et* $\neq *$ *alors* $A_i = 1$ *:*
- *Si* $O_{1_i} = O_{3_i}$ *alors* $B_i += 0$ *mais* $O_{3_i} \neq O_{2_i}$ *donc* $B_i += 1$.
- *Si* $O_{1_i} \neq O_{3_i}$ *alors si* $O_{3_i} = *$ *alors* $B_i = 1/2 + 1/2$ *ou* $B_i = 1 + (0$ *ou* $1)$

$\Longrightarrow B_i = 1$ *ou* 2. *Dans ce cas,* $A_i \leq B_i$.

Si $O_{1_i} \neq O_{2_i}$ *et l'un des deux vaut *, comme la distance est symétrique, posons* $O_{1_i} = *$ *alors* $A_i = 1/2$ *:*

- *Si* $O_{1_i} = O_{3_i}$ *alors* $B_i += 0$ *mais* $O_{3_i} \neq O_{2_i}$ *donc* $B_i += 1/2$.
- *Si* $O_{1_i} \neq O_{3_i}$ *alors* $B_i = 1/2$ *+ (0 ou 1).*
- *Si* $O_{3_i} = *$ *alors* $B_i = 1/2$.

$\Longrightarrow B_i == 1/2$ *ou* 1. *Dans ce cas* $A_i \leq B_i$.

En conclusion, quelques soient les objets O_1, O_2, O_3 :
$\forall i\ A_i \leq B_i$
$\sum_i A_i \leq \sum_i B_i$
$A \leq B$
$\sum_i d_\sigma(O_{1_i}, O_{2_i}) \leq \sum_i d_\sigma(O_{1_i}, O_{3_i}) + \sum_i d_\sigma(O_{3_i}, O_{2_i})$
$d_v(O_1, O_2) \leq d_v(O_1, O_3) + d_v(O_3, O_2)$

5.6.5.3 Vérification de la qualité de centre

Dans cette section, nous vérifions que le centre défini en section 5.6.5.1 respecte les propriétés fondamentales du centre par rapport à la distance décrite en section 5.6.5.2.

Théorème 1 *La fonction* $D(Q, X) = \sum d_v(Q, X_i)$ *est minimisée par le centre défini par la définition 18.*

Preuve 4 *Soit* n *le nombre d'objets du cluster* X, *soit* m *le nombre d'attributs d'un objet. Soit* d_v *notre distance définie précédemment comme étant* $d_v = \sum_i d_\sigma(O_i, C_i)$. *Soit* Q *un centre du cluster :*

$$\sum_{j=1}^{n} d_v(X_j, Q) = \sum_{j=1}^{n} \sum_{i=1}^{m} d_\sigma(X_{j_i}, q_i)$$

$\forall i \in [1, m] \sum_{i=1}^{m} d_\sigma(X_{j_i}, q_i) \geq 0$
Posons n_{q_i} *la fréquence de l'attribut i défini comme représentant pour le centre. La distance entre * et un attribut est toujours de 1/2.* n_{q_i} *étant la fréquence d'apparition de la valeur* q_i *du* $i^{ème}$ *attribut choisi pour être le représentant du centre. Comme* n_{q_i} *est la fréquence d'apparence de* q_i *:* $n - n_{q_i} \geq 0$. *:* $n - n_{q_i} \geq 0$.
Il faut donc que chaque élément de la somme soit minimal. Pour chaque $d_\sigma(X_{i_j}, q_j)$ *deux cas sont possibles :*
Soit le $i^{ème}$ *attribut du centre est déterminé. Nous considérons les attributs du centre ayant une fréquence supérieure à la moitié des voix. Comme* $d(X_{j_i}, q_i) = n - n_{q_i}$ *et* $n_{q_i} \geq \frac{n}{2}$ *alors* $n - n_{q_i} \leq \frac{n}{2}$ *et* $n - n_{q_i}$ *est plus petit que* $\frac{n}{2}$ *par construction. Donc* $d(X_{j_i}, q_i)$ *obtenue par vote majoritaire est minimale.*
Soit le $i^{ème}$ *attribut du centre est indéterminé. Il n'existe aucune valeur majoritaire pour cet attribut. Donc* $\forall n_{q_i}, n - n_{q_i} \geq n/2$ *alors* $d(X_{j_i}, q_i)$ *obtenue avec * est minimale.*
Notre élection de centre minimise la distance intra-cluster.

5.6.6 Expérimentations

Dans le cadre de cette étude, nous réaliserons, dans un premier temps, une étude du comportement de notre algorithme CHyGA en terme de convergence sur les différents jeux de données

Nom	# Inst. (n)	# Attr. (l)	k	Données Manquantes (%)	Instances par classes
AD_5_2	250	2	5	0	50 (chaque)
AD_4_3	402	3	4	0	100 (chaque)
Ruspini	75	2	4	0	20, 23, 17, 15
Iris	150	4	3	0	50 (chaque)
Cancer	683	9	2	0	444, 239
Diabetes	768	8	3	0	500, 268
Lung	32	57	3	4,13	9, 13, 10
Vote	435	16	2	0,22	267, 168
Breast Cancer	286	9	2	0,34	201, 85

TAB. 5.5 – Bases de données utilisées.

présentés en annexe (voir annexe D) dont nous rappellerons les principales caractéristiques dans le tableau 5.5. Ces jeux de données, étant à l'origine des jeux de classification, nous indiquerons le nombre de classes, quand il est disponible, et la répartition d'origine des instances dans les classes. Certains jeux de données comportent des données manquantes et nous indiquerons le pourcentage de données manquantes contenues par chaque jeu de données. Nous comparerons ensuite les résultats de notre algorithme sur ces jeux de données par rapport à des algorithmes classiques de clustering présentés précédemment. Les algorithmes servant pour la comparaison ont été implémentés en Java. Les méthodes hiérachiques donnent comme sortie un dendogramme (voir 5.3.1). Pour pouvoir comparer les résultats, nous indiquons le nombre de clusters souhaité à la méthode et la production du dendogramme est stoppée lorsque ce nombre est obtenu.

5.6.6.1 Présentation des expérimentations

Pour pouvoir comparer notre méthode à d'autres méthodes existantes dans la littérature, nous utilisons une série d'indicateurs statistiques. Les critères choisis sont l'inertie inter-cluster et intra-cluster, définis ci-après, ainsi que le critère d'optimisation de l'algorithme génétique : le critère de Calinksi et Harabrasz défini paragraphe 5.6.2.
Soit $P = \{C_1, C_2, ...\}$ l'ensemble de clusters généré par la méthode prise en compte, soit n le nombre d'instances, soit G le centre de gravité global des données, soit G_c le centre de gravité d'un cluster. On définira l'inertie interne d'un cluster C par :

$$I(C) = \sum_{i \in C} \frac{1}{n} \times d(x_i, G_C)$$

où d est la distance définie, x_i la i$^{\text{ème}}$ instance. Plus $I(C)$ sera faible plus le cluster sera compacte. On utilisera l'inertie intra-cluster : $Intra = \sum_{C \in P} I(C)$ et l'inertie inter-cluster : $Inter = \sum_{C \in p} \frac{n_c}{n} \times d(G_C, G)$ avec n_c le nombre d'instances dans le cluster C.

5.6.6.2 Résultats

Dans un premier temps, nous allons étudier le comportement de notre algorithme CHyGA en terme de robustesse. Nous avons réalisé, pour chaque jeu de données, des statistiques descriptives (voir tableaux 5.6 et 5.7) des résultats trouvés lors de 10 exécutions sur le même jeu de données. Nous indiquons pour chaque jeu de données la moyenne sur les exécutions du meilleur individu (Moyenne), le meilleur sur les 10 exécutions (Max), l'écart type entre les différentes exécutions et la médiane. Nous indiquons également pour chaque jeu de données de classification, la valeur du critère (CH) obtenu à partir de la classification donnée.

Dans toutes nos expérimentations, CHyGA résout un double problème : trouver le nombre de clusters et une bonne répartition des objets entre ces clusters par rapport au critère de Calinski et Harabrasz. Nous pouvons remarquer que pour tous les jeux de données réels, le nombre de clusters trouvés par CHyGA est identique au nombre de classes du jeu de données.
La figure 5.13 nous montre la répartition des objets de la base de données Ruspini après un clustering réalisé par CHyGA. Nous avons utilisé Ruspini qui n'a que deux attributs car il est facile de visualiser les résultats et de noter leur pertinence par rapport à leur répartition dans l'espace en deux dimensions. Nous remarquons que pour cette base de données, CHyGA est stable et robuste : il nous donne toujours la même solution et cette dernière (figure 5.13) montre une bonne séparation des solutions dans le plan.
Nous pouvons remarquer que CHyGA donne comme résultat de son clustering une valeur de critère identique ou meilleure à celle obtenue en utilisant la classification de r{eférence donnée par le jeu de données. Il est évident que notre algorithme, pour qu'il puisse être considéré comme bon, doive trouver une valeur de critère au moins aussi bonne que la valeur CH indiquée dans les tableaux 5.6 et 5.7. Nous pouvons remarquer que dans certains cas, CHyGA trouve un meilleur partitionnement pour le critère désigné que celui indiqué dans le jeu de classification. Cela peut s'expliquer par le fait que les labels de classes donnés par l'expert ne tiennent par compte de tous les attributs de la même façon contrairement à la notion de distance utilisée.
Nous comparons également notre algorithme génétique à celui de Bandyopadhyay [BM01] pour les jeux de données numériques : VGA-clustering. Le tableau 5.8 donne une comparaison des résultats en terme de nombre de clusters trouvés pour des jeux de données numériques utilisés par Bandyopadhyay et Maulik. Nous pouvons observer que CHyGA trouve le nombre correct de clusters pour tous les jeux de données ce que VGA obtient également en utilisant l'indice I. Les autres indices DB, v_D, v_{33} et v_{53} utilisés comme fonction d'évaluation par VGA sont incapables de trouver le nombre correct de clusters.

Puis, dans un deuxième temps, nous comparons notre méthode aux méthodes classiques de la littérature : Kmeans, Kmédioïd, Single link et Complete link. Cette comparaison se fait uniquement sur des jeux de données numériques. En effet, nous n'avons pas implémenté notre méthode de centre partiel et la distance qu'il induit pour les méthodes classiques dédiées aux données numériques.

Pour chaque méthode autre que CHyGA, nous devons indiquer le nombre de clusters désiré car ces méthodes ne déterminent pas le nombre de clusters "optimal" d'elles-mêmes. Dans un soucis d'équité pour la comparaison, nous avons décidé de fixer le nombre de clusters d'entrée des cinq méthodes autres que CHyGA au nombre de clusters de la meilleure solution donnée par CHyGA. Pour les méthodes non déterministes (Kmeans, Kmédioïd), nous avons exécuté au moins dix fois

Base de données numérique		
AD_5_2 (250 Instances, 2 Attributes)		
CH = 387,749390		
	Critère	# cluster
Moyenne	386,20	5
Maximum	387,74	5
Ecart type	0,18	0
Médiane	386,44	5
AD_4_3 (400, 3)		
CH = 3207,414795		
	Critère	# cluster
Moyenne	3207,41	4
Maximum	3207,41	4
Ecart type	0	0
Médiane	3207,41	4
Ruspini (75, 2)		
CH = 425,327362		
	Critère	# cluster
Moyenne	425,32	4
Maximum	425,32	4
Ecart type	0	0
Médiane	425,32	4
Iris (150, 4)		
CH = 486,320984		
	Critère	# cluster
Moyenne	526,42	3
Maximum	559,25 (3)	4
Ecart type	11,78	0,82
Médiane	527,83	3
Cancer (683, 9)		
CH = 912,201294		
	Critère	# cluster
Moyenne	1025,81	2
Maximum	1025,81	2
Ecart type	0	0
Médiane	1025,81	2
Diabetes (768, 8)		
CH = 24,299105		
	Critère	# cluster
Moyenne	1136,18	3
Maximum	1142,49	3
Ecart type	6,98	0
Médiane	1139,2	3

TAB. 5.6 – Statistiques descriptives des résultats de Chyga sur les différents jeux de données numériques. Pour chaque jeu de données, nous indiquons en dessous de sa description, la valeur du critère sur la classification d'origine (CH).

Base de données Nominales		
Lung (32,57) **CH = 9,609707**		
	Critère	# cluster
Moyenne	23,38	2
Max	23,68	2
Ecart type	0,14	0
Médiane	23,34	2
Vote (435,16) **CH = 455,865845**		
	Critère	# cluster
Moyenne	508,79	2
Max	522,34	2
Ecart type	9,76	0
Médiane	507,52	2
Breast Cancer (286,9) **CH = 212,251999**		
	Critère	# cluster
Moyenne	243,63	2
Max	247,49	2
Ecart type	2,99	0
Médiane	243,91	2

TAB. 5.7 – Statistiques descriptives des résultats de CHyGA sur les différents jeux de données nominales. Pour chaque jeu de données, nous indiquons en dessous de sa description, la valeur du critère sur la classification d'origine (CH).

Jeux de données	**# classes**	**# Clusters**					
		CHyGA		**VGA clustering**			
		CH	DB	v_D	v_{33}	v_{53}	$\mathcal{I}(K)$
AD_5_2	5	5	5	4	4	4	5
AD_4_3	4	4	4	2	4	4	4
Iris	3	3	2	2	2	2	3
Cancer	2	2	2	2	2	2	2

TAB. 5.8 – Comparaison de CHyGA et de VGA avec différents indices.

Jeux de données	Critère	CHyGA	Kmeans	Kmed.	Single Link	Complete Link
AD_5_2	Intra	**5,06**	5,17	6,08	93,90	5,62
	Inter	218,85	218,80	218,02	**218,89**	218,88
	Calinski	**387,74**	386,58	163,92	11,74	6,25
	σ	0	28,84	24,68	-	-
	k	-	5	5	5	5
AD_4_3	Intra	**6,54**	6,63	8,75	6,55	19,67
	Inter	410,88	410,88	411,10	**410,83**	410,99
	Calinski	**3207,41**	3206,68	1720,86	3190,08	0,57
	σ	0	385,02	313,31	-	-
	k	-	4	4	4	4
Ruspini	Intra	**38,75**	**38,75**	54,18	40,57	162,15
	Inter	19782,42	19782,42	**19787,28**	19783,07	19782,52
	Calinski	**425,32**	**425,32**	263,92	422,40	2,63
	σ (1)	0	161,56	74,2	-	-
	k (2)	-	4	4	4	4
Iris	Intra	**2,36**	**2,36**	2,86	2,69	2,50
	Inter	46,6264	46,6264	**46,93**	46,62	46,62
	Calinski	**559,26**	**559,26**	322,92	343,85	42,35
	σ (1)	11,78	1,98	108,55	-	-
	k (2)	-	3	3	3	3
Cancer	Intra	14,14	17,08	**13,42**	28,16	20,03
	Inter	235,31	235,31	**237,68**	235,17	235,34
	Calinski	1025,81	1008,74	**1201,66**	298,09	12,06
	σ	0	0	385,02	-	-
	k	-	2	2	2	2
Diabetes	Intra	**246,56**	247,44	276,17	454,01	264,00
	Inter	105763,16	105763,25	105736,98	**105763,90**	105760,35
	Calinski	**1142,39**	133,97	276,17	20,19	0,95
	σ (1)	6,98	17,56	47,51	-	-
	k (2)	-	3	3	3	3

TAB. 5.9 – Meilleurs résultats des différentes méthodes sur les jeux de données classiques de l'UCI. (1) L'écart type, σ, est ici donné uniquement pour les méthodes non déterministes (CHyGA, K-means, K-médïoid). (2) Le nombre de clusters, k, donné aux méthodes ne pouvant pas le déterminer est celui du meilleur résultat connu pour CHyGA.

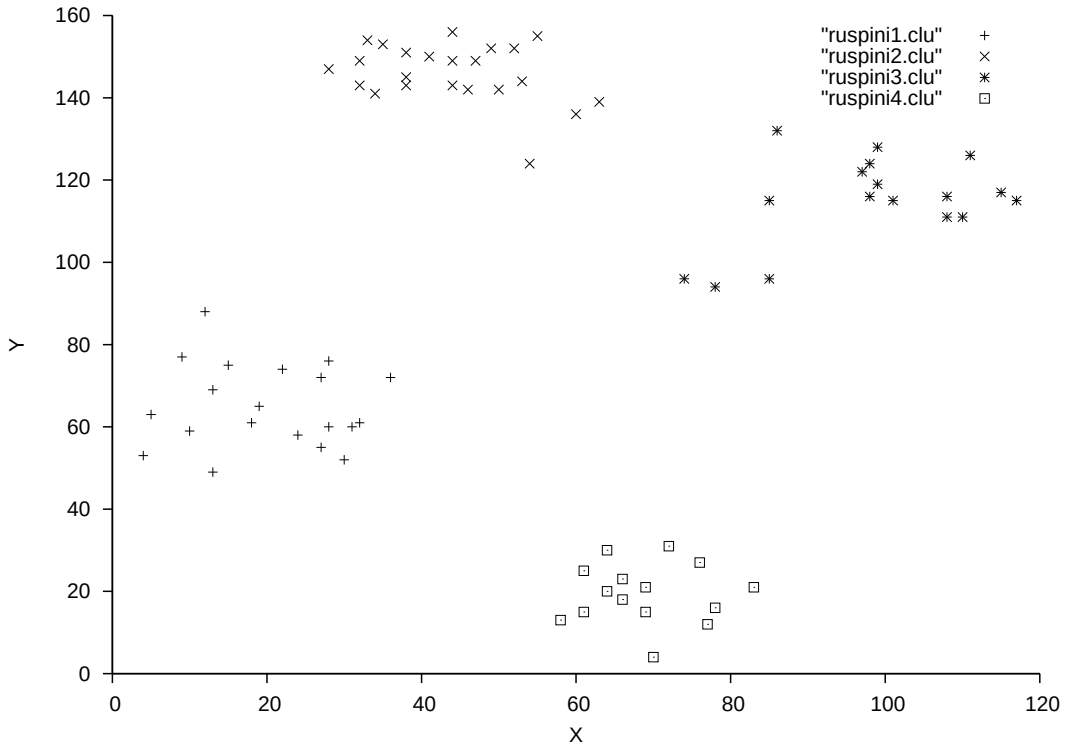


FIG. 5.13 – Exemple de partitionnement obtenu par CHyGA sur la base de données Ruspini.

l'algorithme. Nous indiquons pour chaque critère, la valeur de la meilleure solution trouvée ainsi que l'écart type entre les différentes solutions pour le critère de Calinski. Dans le cadre de la tâche de clustering, l'objectif est de minimiser la distance intra-cluster et de maximiser la distance inter-cluster. Le critère de Calinski est aussi un critère à maximiser.

Pour les jeux de données numériques, nous pouvons remarquer que notre méthode trouve, pour chaque jeu de données, la solution de meilleur critère de Calinski et elle est parfois égalée par Kmeans. Pour le critère intra (distance intra-cluster) qui mesure la compacité du cluster, nous remarquons que CHyGA trouve, pour chaque jeu de données, la meilleure valeur pour ce critère et elle est égalée pour deux jeux de données, Ruspini et Iris, par Kmeans. Pour le critère inter (distance inter-cluster) qui mesure l'écartement des clusters obtenus, nous remarquons que bien que CHyGA ne trouve jamais la meilleure valeur du critère, la valeur trouvée par notre algorithme est très proche de la meilleure trouvée. En effet, si l'on regarde l'erreur commise par CHyGA par rapport à la meilleure valeur du critère trouvée, celle-ci est inférieure à 0,02% pour la base Ruspini, 0,66% pour la base Iris et 0,0006% pour la base Diabetes.

Le jeu de données concernant le cancer du sein (Cancer) a, quant à lui, posé quelques problèmes à notre algorithme. En effet, la méthode de Kmédioïd est capable de trouver une solution de meilleure qualité que CHyGA mais cette solution n'a été obtenue qu'une seule fois et cette méthode se montre très peu robuste sur ce jeux de données comme le montre son fort écart type (385,02).

Pour les trois méthodes stochastiques, nous avons indiqué la valeur de l'écart type entre les solutions pour dix exécutions de la méthode. Nous pouvons remarquer que CHyGA a le plus faible

écart type sauf pour le jeu de données Iris où Kmeans se montre plus robuste.
En terme de temps de calcul, il est évident que notre méthode est plus longue qu'un simple Kmeans puisque nous utilisons une version simple de cet algorithme en tant que recherche locale dans le processus d'hybridation. Les méthodes Single link et Complete link sont plus rapides que CHyGA sur les petits jeux de données mais plus longues sur des grands jeux de données (Diabetes).

5.7 Conclusion

Nous avons présenté dans cette partie, une approche évolutionnaire hybride, CHyGA, permettant de réaliser un clustering avec un nombre de clusters non fixé. Nous avons proposé un codage spécifique d'une solution permettant de traiter des partitionnements de différentes tailles et de facilement accéder à l'information. Nous utilisons l'hybridation avec une recherche locale très utilisée en clustering : Kmeans. Pour mettre en place cette hybridation, nous avons défini une nouvelle notion de centre de cluster, le centre partiel, et nous avons défini une distance pouvant prendre en compte les particularités de ce centre.
CHyGA a été testé sur différentes bases de données et nous avons pu observer sa robustesse en terme de qualité de solutions trouvées et son efficacité pour déterminer le nombre de clusters. CHyGA trouve en effet un nombre de clusters similaire au nombre de classes pour les jeux de données issues de classification. Nous avons ensuite comparé notre méthode à des algorithmes classiques de la littérature Kmeans, Kmédioïd, Single link, Complete link et Average link. Notre méthode se montre la meilleure pour optimiser le critère de Calinki et Harabrasz et la distance intra-cluster et elle est parfois égalée par l'algorithme Kmeans. Pour le critère de distance inter-cluster, nous remarquons que d'autres méthodes obtiennent des meilleurs résultats mais très proches de ceux obtenus par CHyGA. De plus, aucune des méthodes que nous avons utilisées ne détermine le nombre "optimal" de clusters contrairement à CHyGA et nous devons leur indiquer un nombre de clusters présumé. Nous pensons donc que notre méthode est intéressante car elle permet de travailler sans avoir à indiquer le nombre de clusters ce qui est particulièrement intéressant dans le cas de données réelles.
Dans le cadre de cette étude nous nous sommes comparés à des méthodes classiques de la littérature : Kmeans, Kmédïoid, Single Link, Complete Link, Average Link. Ces méthodes ainsi que notre algorithme génétique ont été mis en ligne pour les utilisateurs du Centre d'Informatique et de Biologie de Lille (CIB) et de la génopole pour qu'ils puissent directement les utiliser à travers un serveur en chargeant leurs données.
Nous n'avons pas pu comparer notre algorithme sur des jeux de données nominaux. Il faudrait pour cela que nous puissions prendre en compte dans les méthodes classiques les notions de centre partiel et de sa distance associée que nous avons mis en œuvre.

Le critère de qualité, utilisé par notre algorithme génétique, est une aggrégation de plusieurs critères et il nous semble intéressant de réaliser une modélisation mutlicritère de ce problème.

Chapitre 6

Méthodes d'optimisation pour les règles d'association

Dans le cadre de nos travaux, nous nous sommes intéressés aux règles d'association qui constituent une des tâches de la fouille de données. Dans le cadre de ma thèse, nous avons voulu développer une approche par métaheuristique de la recherche de règles d'association. Nous avons, dans un premier temps, essayé d'identifier les limites des algorithmes classiques de la littérature et notamment de l'algorithme Apriori que nous détaillerons et auquel nous nous comparerons.
Nous avons essayé de faire en sorte que notre démarche soit la plus générale possible dans la conception du modèle de l'algorithme génétique que nous avons développé de manière à l'adapter facilement à différents champs d'application et à y intégrer facilement des connaissances du domaine considéré.
Nous donnerons tout d'abord une définition générale de la recherche de règles d'association et indiquerons le vocabulaire nécessaire à la compréhension de ce problème. Nous donnerons ensuite les principaux algorithmes permettant cette recherche en insistant plus particulièrement sur l'algorithme Apriori et ses limites. Nous détaillerons ensuite l'algorithme génétique que nous avons spécialement développé pour la recherche de règles d'association : ASGARD. Nous présenterons tout d'abord quelques critères de qualité existant dans la littérature et les analyserons pour nous permettre de choisir celui qui nous servira de fonction d'évaluation. Nous présenterons ensuite le codage utilisé en rappelant les différentes représentations possibles. Le choix de la représentation que nous avons effectué a impliqué l'utilisation d'opérateurs spécifiques et notamment de plusieurs mutations dont nous donnerons les particularités. La mise en œuvre de plusieurs mutations nous a amené à travailler sur la mise en place de taux d'application adaptatifs des opérateurs dont nous détaillerons les principes.
Nous appliquerons ensuite notre méthodologie à une base de données classique de l'UCI : Nursery school puis l'adapterons enfin à la problématique biologique expliquée au chapitre 3 dans un contexte d'apprentissage supervisé. Nous présenterons également les outils de visualisation de règles d'association mis à disposition des utilisateurs.

6.1 La recherche de règles d'association

La recherche de règles d'association est une tâche importante dans la fouille de données et est largement étudiée [AIS93, MN99, Sch01]. Ce problème consiste à découvrir des règles du type *IF C THEN P* où C est la condition de la règle et P sa prédiction. La condition est une conjonction de termes : $term_1 \wedge term_2 \wedge ... \wedge term_n$. Chaque terme est du type : $< attribut\ opérateur\ valeur >$. Dans notre cas, l'attribut est de type catégorique et la valeur appartient à un domaine.
Pour cette étude, nous nous intéressons plus particulièrement aux règles d'association sous forme conjonctive dont la partie gauche (C) est une conjonction d'un ou plusieurs tests d'égalité. La condition C sera composée d'attributs distincts.

Les règles sont testées sur les exemples contenus dans la base de données que l'on nomme instances.
Nous utiliserons les notations suivantes :

- $|C|$: nombre d'instances recouvrant la partie C de la règle d'association,
- $|P|$: nombre d'instances recouvrant la partie P de la règle,
- $|C\&P|$: nombre d'instances recouvrant en même temps les parties C et P,
- N : nombre total d'instances de la base.

La confiance d'une règle d'association correspond à la fréquence d'occurences de P lorsque C est observé : $b(C,P) = \frac{|C\&P|}{|C|}$. Elle représente la justesse de la règle : si la confiance est forte cela signifie que la règle est peu contredite par les instances de la base.

Le support d'une règle d'association correspond à la fréquence de la règle (fréquence de la conjonction des descriptions C et P) : $s(C,P,N) = \frac{|C\&P|}{N}$. Il représente le degré de généralité d'une règle. La fréquence de prédiction sera notée $a(P,N) = \frac{|P|}{N}$. P peut dans certains cas être considéré comme une classe et on parlera alors de fréquence de classe.
L'objectif est de découvrir les règles les plus pertinentes, les plus explicatives. Différentes mesures de qualité peuvent être utilisées. Ces mesures peuvent être vues comme des fonctions à optimiser (maximiser la qualité, minimiser les erreurs, ...). De plus, la combinatoire associée au problème, de par le choix des attributs à faire figurer dans la règle et leurs différentes valeurs possibles, font que la taille de l'espace de recherche des règles candidates est exponentielle.
Le problème de recherche d'association est NP-complet [AIP01]. Anguilli montre que le problème se réduit en un problème de CLIQUE qui a été démontré NP-complet [GJ79].

6.1.1 Algorithme Apriori

Dans cette partie, nous allons décrire l'algorithme Apriori développé par Agrawal en 1994 [AS94]. Nous essaierons de voir les différentes propriétés de cet algorithme et ses limitations dans le cadre de notre travail.

L'algorithme Apriori permet d'**extraire toutes les règles dont le support et la confiance sont supérieurs aux seuils MINSUP et MINCONF fixés arbitrairement**.
Il se divise en deux parties :

- la *recherche des itemsets fréquents* : les itemsets fréquents sont les groupes d'items (attri-

buts) dont le support est supérieur à **MINSUP**,
- l'*extraction des règles* parmi l'ensemble des itemsets fréquents : les règles dont la confiance est supérieure à **MINCONF**.

Recherche des itemsets fréquents
C'est à cette partie que l'on donne le nom d'algorithme **Apriori**. Pour construire l'ensemble des itemsets fréquents, l'algorithme 9 construit des ensembles de plus en plus grand d'itemsets fréquents.

Algorithme 9 Algorithme de recherche des itemsets fréquents

$L_1 = \{1\text{-itemsets fréquents}\}$
$k \leftarrow 2$
while $L_{k-1} \neq \phi$ **do**
  $C_k =$ apriori_gen(L_{k-1})
  for all $instances \quad t \in T$ **do**
    $C_t =$ subset(C_k,t)
    for all candidats $c \in C_t$ **do**
      c.count++ ;
    end for
  end for
  L_k = { c $\in C_k$ / c.count $\geqslant$ MINSUP }
  k++
end while
L = $\cup_i L_i$

De manière itérative, pour chaque parcours de la base de données, les itemsets fréquents de taille k (k-itemsets noté L_k) sont calculés à partir des itemsets fréquents de taille $k-1$. Le procédé est le suivant :
- construire un ensemble candidat C_k, jointure de L_{k-1} avec L_{k-1}, en éliminant les k-itemsets (itemsets de taille k) qui contiennent un sous-itemset non fréquent.
Le **principe fondamental** sous-jacent est que **tout sous-itemset d'un k-itemset fréquent est fréquent** (Si ABC est fréquent, alors A, B, C, AB, AC, BC le sont nécessairement) (monotonie du support).
- calculer le support de chaque itemset candidat, et ne retenir que ceux dont le support est supérieur à **MINSUP**.

Cette partie de l'algorithme est **la plus coûteuse, à cause de la génération des candidats**. En effet, pour p items de départ, la taille de l'espace de recherche des n-itemsets fréquents (obtenue pour **MINSUP** = 0) vaut C_n^p, soit au total 2^n.

Deuxième partie : extraction de règles
L'extraction des règles n'est pas une notion intrinsèque de l'algorithme Apriori. La confiance est généralement utilisée après la recherche des itemsets fréquents, car elle définit une relation simple avec le support. La structure de données de l'algorithme Apriori prévoit donc d'enregistrer, à chaque passe, le support des k-itemsets fréquents.
L'algorithme 10 présente l'extraction des règles par la confiance.

Algorithme 10 Algorithme d'extraction de règles

for chaque itemset fréquent l **do**
  générer tous les sous-itemsets non vides s de l
end for
for chaque sous-itemset non vide s de l **do**
  produire la règle s $\Rightarrow$ (l-s) si $\frac{support(l)}{support(s)} \geqslant$ MINCONF
end for

La recherche de règles d'association se déroule donc en deux parties :

- une opération longue et coûteuse : le calcul de l'ensemble L des itemsets fréquents, l'algorithme Apriori en lui-même.
- une opération rapide : l'extraction des règles dans l'ensemble L des itemsets fréquents.

La conclusion à tirer de l'étude du fonctionnement est que **la confiance n'est pas une notion inhérente de l'algorithme Apriori**, l'extraction des règles pouvant se faire par tout autre critère basé sur le support des itemsets fréquents.
Peut-on remplacer le support par un autre critère ? Nous reprendrons les observations et conclusions faites lors du stage qu'a effectué F. Blondel dans notre équipe [Blo02].

Principe fondamental et mesure de qualité d'une règle pour Apriori
Soit $\mathcal{I}$ un ensemble d'items, $\mathcal{T}$ un ensemble de transactions et C une mesure. Le principe fondamental donne une condition nécessaire sur la mesure (ici le support) :

$$\forall X, Y \subset \mathcal{I}, \forall z \in \mathcal{I}, X \subset Y, C(X \Rightarrow z) \geqslant C(Y \Rightarrow z)$$

Remarque : la condition porte sur les règles et non plus sur les groupes d'items. On peut toutefois se ramener au cas du support en posant $Supp(X \Rightarrow z) = Supp(Xz)$

Inconvénient du support
Dans la suite, on considère un ensemble d'items A, B et Z et on cherche à évaluer la pertinence des règles ayant Z comme prédiction (A=>Z, B=>Z ou AB => Z).
Un item est présent dans la transaction s'il vaut un, absent s'il vaut zéro.

CONSEQUENCE : le support aura tendance à évincer les règles "rares", mais pouvant être intéressantes au niveau de la confiance.
EXEMPLE :

A	B	Z	
0	0	1	$\times 3$
0	0	0	$\times 3$
1	0	1	$\times 2$

MINSUP = 30%

$$Supp(A) = \frac{2}{8} = 25\%$$

$$Supp(B) = 0$$

Apriori s'arrête car il n'a trouvé aucun itemset fréquent.
Pourtant, $Conf(A \Rightarrow Z) = 100\%$, donc la règle $A \Rightarrow Z$ qui n'a pas été trouvée peut être intéressante !
Une solution envisageable pour rendre le support moins contraignant serait les règles d'association multi-niveaux [Maa] : le support diminue au fur et à mesure des passes, afin de favoriser les règles contenant beaucoup d'attributs.

Conséquence sur la recherche de règles d'association
Apriori est un algorithme correct (il ne trouve que les règles ayant un support et une confiance supérieurs aux seuils fixés) et complet (il trouve toutes les règles ayant un support et une confiance supérieurs aux seuils fixés).

Observation sur la confiance
Le calcul des règles se fait après le calcul des itemsets fréquents, et c'est un opération peu coûteuse. La notion de confiance n'est donc pas essentielle dans l'algorithme Apriori, et elle pourrait très bien être remplacée par une autre mesure.

6.2 Autres algorithmes de recherche de règles

Il existe de nombreux algorithmes de recherche de règles d'association qui peuvent être des variantes d'Apriori. La figure 6.1 présente une classification non exhaustive des différents algorithmes.

6.2.1 Algorithmes dynamiques

DIC (dynamic itemset counting) a été proposé par Brin et al. en 1997 [BMS97]. Il permet d'améliorer l'efficacité de la recherhce des itemsets fréquents par niveaux en diminuant le nombre de parcours de la base de données. Dès qu'un itemset candidat examiné au niveau k atteint le seuil de fréquence, DIC commence à examiner les itemsets de niveau $k+1$ générés par celui-ci.

6.2.2 Extension d'Apriori séquentiel

Les extensions d'Apriori tentent de pallier à certains de ces défauts.
AprioriTID
AprioriTID est une variante de l'Apriori proposé par Agrawal et al. [AS94]. Il permet de diminuer progressivement la taille de la base de données considérée, dans le but de la stocker en mémoire et de ne plus réaliser d'opération d'entrée / sortie, après le premier parcours de celle-ci.

Algorithme A-Priori Partition
Savasere et al. [SON95] propose une extension d'Apriori où la base de données est divisée en N

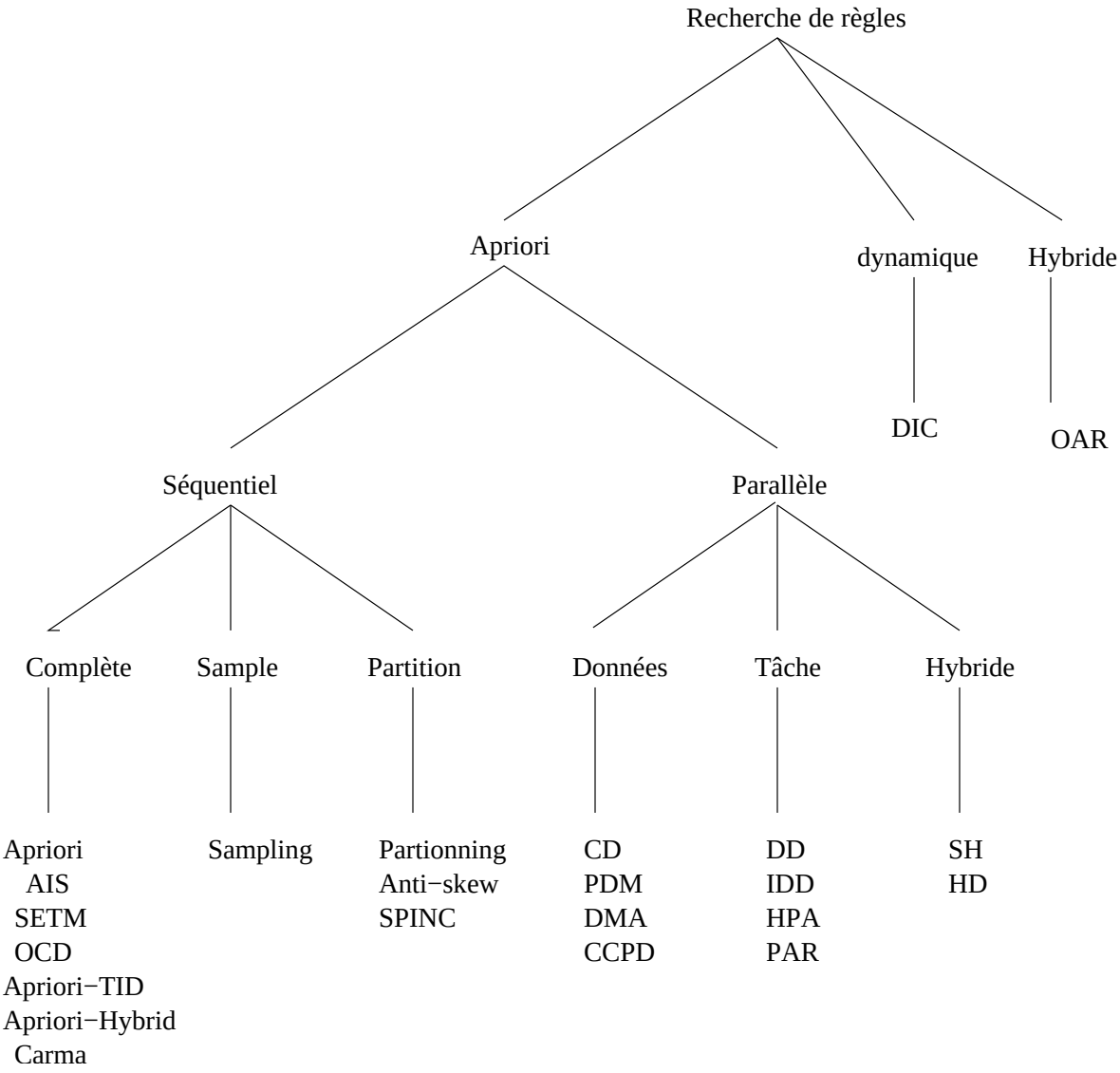


FIG. 6.1 – Classification des algorithmes de recherche de règles complets basés sur les itemsets.

partitions qui tiennent chacune en mémoire. Les partitions peuvent ainsi être facilement traitées en parallèle. Chaque partition est traitée indépendamment et on réalise la découverte des ensembles fréquents pour chaque partition. On remarque qu'un ensemble fréquent doit l'être dans au moins une partition.

6.2.3 Alternatives

Il existe aussi différentes alternatives à l'algorithme Apriori et notamment des travaux sur une représentation sous forme de treillis des itemsets.
Toivonen [Toi96] dans l'algorithme Sampling propose de travailler sur un échantillon de la base qui tienne en mémoire, à partir duquel on construit l'ensemble des itemsets fréquents dans l'échantillon ainsi que sa bordure négative constituée des itemsets non fréquents minimaux au sens où toutes leurs parties sont fréquentes, ce qui limite le risque de non-exhaustivité.
Dans une logique d'exploration du treillis en profondeur et non plus en largeur, d'autres algorithmes ont été proposés, tel que FP-Growth [HPY00] qui utilise une représentation très condensée des données pour évaluer les fréquences par comptage dans la base ou Éclat [ZPOL97] qui les évalue par une procédure accélérée d'intersection des tid-listes.
MaxEclat [ZPOL97] ou Max-Miner [Bay98] sont des algorithmes de recherche d'itemsets fréquents maximaux où les sur-ensembles d'itemsets sont non-fréquents. Mais ils se prêtent mal au calcul du support des ensembles fréquents qu'ils contiennent, calcul pourtant nécessaire à celui de la confiance des règles générées.

6.2.4 Versions parallèles

Il existe de nombreux algorithmes parallèles et distribués basés sur la recherche d'itemsets fréquents et notamment sur l'algorithme Apriori. Dans [Zak99], Zaki propose de classer les algorithmes par stratégie d'équilibrage de charge (load-balancing), architecture et parallélisme. Dans la classification donnée dans la figure 6.1, nous nous concentrons sur le parallélisme utilisé : parallélisme de données et parallélisme de tâche [CDG+97]. Les deux paradigmes diffèrent par le fait que l'ensemble de candidats soit distribué ou non à travers les processeurs. Dans le paradigme de parallélisme de données, chaque noeud compte le même ensemble de candidats alors que dans le paradigme de parallélisme de tâche, l'ensemble des candidats est divisé et distribué à travers les processeurs, et chaque noeud compte un ensemble différent de candidats. La base de données peut théoriquement être divisée dans l'un ou l'autre paradigme. Dans la pratique, pour un I/O plus efficace on suppose habituellement que la base de données est divisée et distribuée à travers les processeurs.
Parallélisme de données :
Les algorithmes qui adoptent le paradigme de parallélisme de données incluent : Count Distribution [AS96], PDM (Parallel Data Mining) [PCY95], DMA (Distributed Mining Algorithm) [CHN+96], et CCPD (Common Candidate Partitioned Database) [ZOPL96]. Ces algorithmes parallèles diffèrent par l'utilisation ou non de techniques d'élimination de candidats (pruning) ou de technique de comptage de candidats efficaces.
Parallélisme de tâches :
Les algorithmes adoptant le paradigme de parallélisme de tâche incluent : DD (Data Distribu-

tion) [AS96], IDD (Intelligent Data Distribution) [HKK97], HPA (Hash-based Parallel mining of Association rules) [SK96] et PAR (Parallel Association Rules) [ZPOL97], qui inclue les algorithmes (Par-Eclat, Par-MaxEclat, Par-Clique et Par-MaxClique). Ils divisent tous aussi bien les candidats que la base de données parmi les processeurs. Ils diffèrent dans la façon dont les candidats et la base de données sont divisés.

6.2.5 Comparaison des différents algorithmes

Soient m le nombre d'articles dans chaque transaction, L_k le plus grand itemset de taille k dans une base de données D, C_k l'ensemble des candidats de taille k et $\overline{C_k}$ l'ensemble des candidats de taille k obtenu lors de la passe précédente. Le tableau 6.1 récapitule les différents algorithmes présentés et leurs spécificités.

Freitas [FLF00] indique que la tâche de ces algorithmes est clairement définie et déterministe car ils doivent tous découvrir le même ensemble de règles, celui qui vérifie les conditions de support et de fréquence qui ont été retenues.
En fait, l'intérêt même des règles générées par ces algorithmes est discutable. En effet, les règles ayant une fréquence inférieure au seuil sont éliminées alors que certaines d'entre elles ayant une très forte confiance sont susceptibles de présenter un réel intérêt, par exemple dans les problèmes de ciblage, en marketing ou en médecine. On peut être ainsi conduit à baisser le seuil de support, mais il y a alors beaucoup trop d'ensembles fréquents et les algorithmes peinent à extraire de telles règles.
Une étude nous a démontré expérimentalement que, pour les données que nous voulons étudier, l'utilisation de tels algorithmes à cause des raisons précitées devenait vite inintéressante [Blo02].
Nous proposerons, dans la suite de ce chapitre, une étude de différents critères de qualité de règle d'association avant de proposer un algorithme pouvant utiliser indistinctement l'un de ces critères.

6.3 Critères de qualité d'une règle d'association

De nombreuses mesures existent pour estimer la qualité des règles d'association de la forme *IF C THEN P*. Nous ne les présentons pas toutes ici. Pour avoir une vue globale, le lecteur peut se référer aux articles de Freitas [Fre99], Tan et al. [TKS02] et aux thèses de S. Guillaume [Gui00] et O. Teytaud [Tey01]. Cette problématique reste une préoccupation, en atteste les articles récents sur le sujet [LA03, Azé03]. Nous présentons ici les mesures les plus fréquemment utilisées par les algorithmes d'optimisation en reprenant les notations prises dans le chapitre 2.

Définition 19 *On parle de mesure symétrique quand la mesure évalue de la même façon IF C then P et IF P then C sinon on parle de mesure dissymétrique.*

6.3.1 Mesures composées

Différents auteurs ont proposé des mesures combinant plusieurs mesures de base. Les plus basiques sont basées sur le support (s) et la confiance (b). D'autres fonctions plus complexes ont

Algorithme	**Nb. de Scan**	**Structure de données**	**Commentaires**
AIS	m+1	Non spécifiée	Adapté pour des transactions de petite cardinalité ; la conséquence ne comporte qu'un item
SETM	m+1	Non spécifiée	Compatible avec SQL
Apriori	m+1	L_{k-1} : Table de hachage C_k : Arbre de Hachage	Pour des transactions de cardinalité moyenne ; Meilleur en terme de résultats que AIS et SETM.
Apriori-TID	m+1	L_{k-1} : Table de hachage, C_k : tableau indexé par les TID, $\overline{C_k}$ Structure séquentielle, ID : bitmap	Très lent lorque le nombre de $\overline{C_k}$ est important ; Meilleur que Apriori quand il y a peu de $\overline{C_k}$.
Apriori-Hybrid	m+1	L_{k-1} : Table de hachage, 1ère Phase : C_k : Arbre de hachage, 2ème phase : C_k : Tableau indexé par IDs, $\overline{C_k}$ structure séquentielle, ID : bitmap	Meilleur que Apriori.
OCD	2	Non spécifiée	Applicable à des grosses bases de données (DB) avec un seuil de support faible.
Partition	2	Table de hachage	Applicable à des grosses bases de données avec une grande cardinalité des données ; favorise la distribution homogène des données.
Sampling	2	Non spécifiée	Applicable à des grosses bases de données (DB) avec un seuil de support faible.
DIC	Dépend de la taille des intervalles		Les candidats de taille croissante sont générés à la fin d'un intervalle.
CARMA	2	Table de hachage	S'applique à des transactions lues à partir d'un réseau.
CD	m+1	Table de hachage et arbre	Parallélisme de données.
PDM	m+1	Table de hachage et arbre	Parallélisme de données ; avec élimination des candidats
DMA	m+1	Table de hachage et arbre	Parallélisme de données ; avec élimination des candidats

TAB. 6.1 – Comparaison de principaux algorithmes d'extraction de règles.

aussi été proposées comme par exemple la mesure de Piatetsky-Shapiro [PS91] qui est l'une des plus populaires :

$$PS(C,P,N) = |C\&P| - \frac{|C||P|}{N} \tag{6.1}$$

Cette mesure tient compte du déséquilibre de la distribution des classes, c'est-à-dire si les instances qui appartiennent à une classe sont plus fréquentes ou plus rares que les instances qui appartiennent aux autres classes. La fonction PS favorise ainsi les règles qui prédisent les classes minoritaires.

Smyth et Goodman [SG91] ont proposé la *Jmeasure* qui est une mesure du degré d'intérêt d'une règle :

$$\begin{aligned} Jmeasure(C,P,N) &= \frac{|C|}{N}\Big(b(C,P) \times log\big(\frac{a(P,N)}{b(C,P)}\big)\Big) \\ &+ (1-b(C,P)) \times log\Big(\frac{1-b(C,P)}{1-a(P,N)}\Big) \end{aligned} \tag{6.2}$$

Cette mesure est d'autant plus grande que l'intérêt de la règle est important. Elle peut être divisée en deux parties : la première $\frac{|C|}{N}\Big(b(C,P) \times log\big(\frac{a(P,N)}{b(C,P)}\big)\Big)$ favorise des règles générales, la seconde dérive de la théorie de l'information. Si on se donne une règle de type *IF C=c THEN P=p,* la $Jmeasure$ compare la distribution a priori de P avec la distribution a posteriori de P étant donné que $C = c$.
Dans le cas de la $Jmeasure$, il est parfois possible de trouver une valeur erronée car la $Jmeasure$ aura également une valeur élevée quand il y a une forte différence entre la distribution a posteriori de P et la distribution a priori de P étant donné C. Le problème résulte du fait que cette différence est calculée de façon symétrique.

Parmi les autres mesures proposées, certaines reposent sur la statistique des exemples et sont symétriques. On citera par exemple le $lift$ ($\frac{P(CP)}{P(C)P(P)}$), le $khi2$ ou le coefficient de corrélation r. D'autres mesures, telle la conviction et l'intensité d'implication font intervenir la statistique des contre-exemples ce qui contribue à les rendre dissymétriques. La conviction [BMS97] est un analogue du lift appliqué aux contre-exemples. Dans la même logique, l'intensité d'implication [Gra79] évalue l'étonnement statistique que provoque le nombre de contre-exemples de la règle, soit l'hypothèse d'indépendance de C et P.
Une énumération de ces critères a été effectuée par A. Wasson et T. Longuemart [WL01] et une étude statistique par M. Khabzaoui [KDNT03].

6.3.2 Comparaison des mesures de qualité

Étant donné la multiplicité des mesures de qualité, certains auteurs [Kod00, PS91] proposent des conditions et des indicateurs pour connaître la qualité des règles.
Pour Piatetsky-Shapiro [PS91], une bonne mesure doit prendre comme valeur :

1. 0 en cas d'indépendance de C et de P.

Mesure	**Incompatibilité** $CP = \varnothing$	**Indépendance** $P(CP) = P(C)P(P)$	**Règle logique** $C \subset P$
Confiance	0	$\frac{\|P\|}{N}$	1
PS	$-\frac{\|C\|\|P\|}{N}$	0	$\|C\| - \frac{\|C\|\|P\|}{N}$
Lift	0	1	$\frac{N}{\|P\|}$
Conviction	$1 - \frac{\|P\|}{N}$	1	∞
Jmesure	$P(C)log\left(\frac{1}{1-P(P)}\right)$	0	$P(P)log\left(\frac{1}{P(P)}\right)$

TAB. 6.2 – Comportement de différentes mesures dans les conditions d'incompatibilité, d'indépendance et de règle logique.

2. > 0, en cas d'attraction, P(CP)> P(C)P(P)
3. < 0, en cas de répulsion, P(CP)< P(C)P(P)

Dans [ZZ02], Zhang et Zhang remplacent les conditions 2 et 3 par des conditions de normalisation :

2bis. = 1, au cas où la règle est logique (confiance égale à 1, soit $C \subset P$)

3bis. = -1, en cas d'incompatibilité (confiance nulle, soit $C \cap P = \varnothing$)

6.3.3 Mise en œuvre dans les algorithmes génétiques

Certains auteurs ont remarqué que, dans les premières générations d'un algorithme génétique utilisant la $Jmeasure$ comme fonction d'évaluation, la qualité des individus était souvent nulle car les règles ne couvraient aucun exemple. Wang et al. [WTL98] proposent une amélioration avec :

$$J1 = \frac{|C|}{N}\Big(b(C,P) \times log\Big(\frac{b(C,P)}{a(P,N)}\Big)\Big) \tag{6.3}$$

Cette formule pose aussi des problèmes car la qualité est très faible au début et montre une convergence très lente vers la meilleure solution. Araujo *et al.* [ALF99] proposent alors :

$$F = \frac{\big(w1 \times J1 + w2 \times (\frac{N_{pu}}{NT})\big)}{w1 + w2} \tag{6.4}$$

où N_{pu} représente le nombre d'attributs utiles de la partie C et NT le nombre total d'attributs apparaissant dans la partie C. Un attribut A apparaissant dans une règle avec une valeur donnée, est dit utile s'il y a au moins une instance dans les données qui possède à la fois, l'attribut A et sa valeur indiquée par la règle, et l'attribut but et sa valeur indiquée par la règle. Cette fonction est composée de 2 termes pondérés par $w1$ et $w2$, le premier terme reprend la métrique $J1$ et le second essaie de pallier à ses défauts. $w1$ doit être plus important que $w2$ pour limiter l'influence du second terme. Les auteurs [ALF99] suggèrent que w_1 et w_2 aient respectivement les valeurs 0,6 et 0,4.

Dans [MT01], les auteurs observent que le terme $\frac{N_{pu}}{NT}$ est constant après un nombre de générations g_n et il est donc inutile, voire coûteux, de le calculer une fois ce nombre atteint. Les auteurs

utilisent cette remarque et définissent la fonction suivante pour la mesure de qualité des règles à une génération g :

$$F'(règle) = \begin{cases} \frac{w1 \times J1 + big(w2 \times \left(\frac{N_{pu}}{NT}\right)\right)}{w1+w2} & \text{si } g < g_n \\ J1 & \text{sinon} \end{cases} \quad (6.5)$$

Le paramètre g_n est fixé expérimentalement en observant à partir de quelle génération, en moyenne, $\frac{N_{pu}}{NT}$ est constant.

6.4 ASGARD : un algorithme de recherche de règles d'association

Le travail présenté dans cette partie a fait l'objet d'une publication dans la revue ECA [JDT02a] et de présentations aux conférences ROADEF 2003 [KJDT03] et ECCB 2003 [JDT03b].
ASGARD[1] (Adaptive Steady state Genetic Algorithm for association Rule Discovery) est un algorithme génétique adaptatif que nous avons développé pour la recherche de règles d'association. Nous définissons tout d'abord sa représentation et la replaçons par rapport à celles existantes dans la littérature. A partir de la représentation que nous avons choisie, nous avons défini des opérateurs répondant au problème de recherche de règles d'association ce qui nous a conduit à utiliser plusieurs opérateurs. Pour pouvoir utiliser ces opérateurs, nous avons mis en place des mécanismes adaptatifs que nous présentons et dont nous montrons l'utilité.

Le schéma général de l'algorithme génétique ASGARD (voir figure 6.2) est celui donnant les meilleurs résultats en terme de robustesse et de temps de convergence parmi les différents schémas qui ont été testés. Celui-ci permet en effet un bon parcours de l'espace de recherche.
Cet algorithme génétique est de type "steady state" et sans doublon. Il travaille en sous-populations. Chaque individu d'une même sous-population a le même attribut but P dont la valeur peut être fixée ou laissée libre. Donc, au sein d'une sous-population, soit tous les individsus ont le même attribut but et la même valeur pour celui-ci, soit les individus ont le même attribut but mais avec des valeurs différentes.

6.4.1 Représentation

Nous avons vu dans le paragraphe 2.4.4 qu'il existe deux types de représentation : l'approche Michigan et l'approche Pittsburg.
Dans notre approche, chaque individu de la population représente une règle candidate. Nous avons choisi pour chaque individu un codage efficace permettant, à moindre coût mémoire, de représenter toutes les règles valides de notre problème. La partie C de la règle pouvant être de taille variable, la taille des chromosomes est variable. Chaque élément représente un terme : un nom d'attribut et sa valeur. Le dernier élément représente toujours l'attribut but de la règle et sa valeur (voir figure 6.3).

[1] NDA : Asgard : lieu où demeurent les dieux de la mythologie nordique (Odin, Thor, ...).

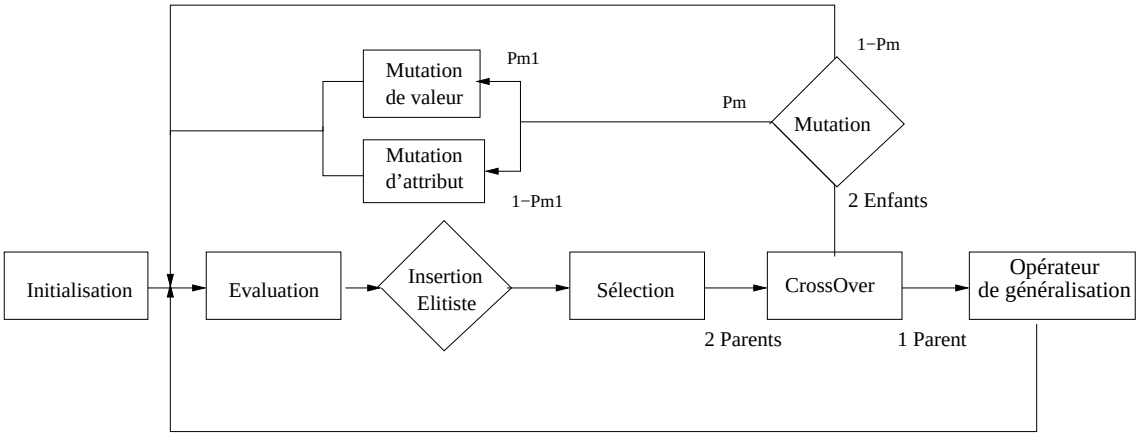


FIG. 6.2 – Le schéma général de l'algorithme génétique ASGARD.

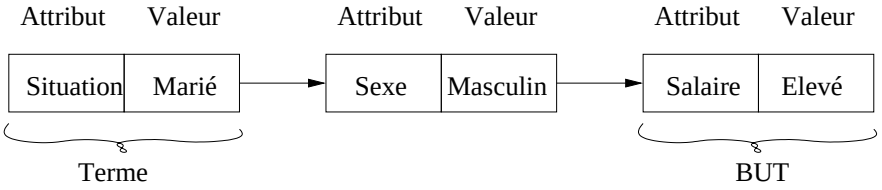


FIG. 6.3 – Exemple de représentation d'un individu de l'algorithme génétique.

6.4.2 Initialisation

La population initiale est générée aléatoirement : pour chaque individu, un nombre de termes est choisi aléatoirement entre 1 et un nombre maximal spécifié par l'utilisateur. Ensuite, l'attribut de chaque terme est choisi aléatoirement parmi l'ensemble des attributs des données (sans doublon). Enfin, la valeur de chaque attribut est choisie aléatoirement dans son domaine. La valeur de l'attribut but peut soit être traitée comme indiqué précédemment, soit être spécifiée par l'utilisateur, ce qui lui permet d'affiner sa recherche.

6.4.3 Fonction d'évaluation (fitness)

Nous avons choisi d'utiliser comme fonction d'évaluation, la fonction F' définie par l'équation 6.5 du paragraphe 6.3.3. Nous avons utilisé les poids $w1$ et $w2$ habituellement fixés dans la littérature (respectivement à 0.6 et 0.4). La valeur de g_n a été fixée expérimentalement suivant le problème traité.

6.4.4 Opérateurs

La reproduction est effectuée à l'aide d'opérateurs (croisement, mutation, ...). Tous les individus d'une même sous-population ont le même attribut but. Les opérateurs opèrent uniquement sur

les individus d'une même sous-population.

Croisement (Crossover)

L'opérateur de croisement (recombinaison) combine le matériel de plusieurs parents pour obtenir un ou plusieurs enfants. Nous avons voulu que notre opérateur de croisement conserve le plus de blocs d'information possibles. Ainsi dans notre cas, l'opérateur de croisement a deux fonctionnements différents selon que les individus parents possèdent ou non des termes communs.
Dans le cas où les individus parents $P1$ et $P2$ ont un ou plusieurs attributs en communs dans la partie C, on choisit aléatoirement un attribut en commun et on échange la valeur de cet attribut entre $P1$ et $P2$ (Voir figure 6.4).
Dans le cas contraire où $P1$ et $P2$ n'ont pas d'attribut en commun, on choisit aléatoirement un terme de $P1$ et on l'insère dans $P2$ avec une probabilité inversement proportionnelle à la longueur de $P2$. De la même façon, on tente d'insérer un terme de $P2$ dans $P1$ (Voir figure 6.5).

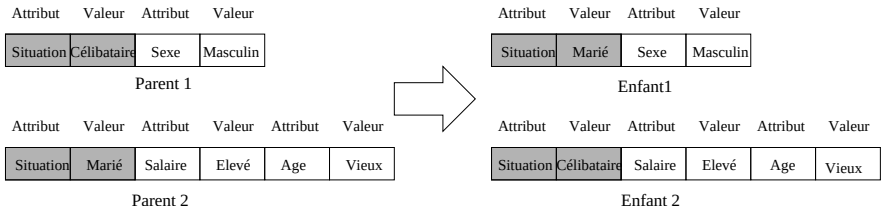


FIG. 6.4 – Exemple de croisement dans le cas 1.

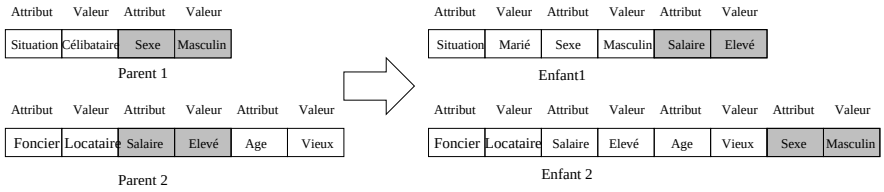


FIG. 6.5 – Exemple de croisement dans le cas 2.

Mutations

Notre algorithme génétique possède deux types de mutation gardant un nombre de termes constant : par attribut ou par valeur. Les individus mutés sont gardés s'ils sont meilleurs que le moins bon individu de la population (élitisme) et s'ils ne sont pas déjà présents, car nous ne voulons pas de doublon.

Mutation par valeur

L'opérateur choisit aléatoirement un attribut et en modifie sa valeur en choisissant aléatoirement une autre valeur du domaine.
Par exemple, pour l'individu : *(situation=célibataire)* $\wedge$ *(sexe=Masculin)*, la mutation peut choisir aléatoirement l'attribut *situation* et la valeur *marié*.
Le nouvel individu obtenu est alors : *(situation=marié)* $\wedge$ *(sexe=Masculin)*.

Mutation par attribut
L'opérateur choisit aléatoirement un terme et remplace son attribut par un autre déterminé aléatoirement. La valeur du nouvel attribut est choisie aléatoirement dans son domaine.
Par exemple, considérons l'individu : *(situation=célibataire)* $\wedge$ *(sexe=Masculin)*, la mutation choisit aléatoirement le terme *(situation=célibataire)* et le remplace par *(salaire=élevé)*.
Le nouvel individu obtenu est alors : *(salaire=élevé)* $\wedge$ *(sexe=Masculin)*.

Opérateur de généralisation
L'objectif de la méthode est de découvrir des règles simples qui ne contiennent pas beaucoup de termes. Nous mettons donc en œuvre un opérateur de généralisation qui supprime un terme avec une probabilité proportionnelle au nombre de termes Nb_{termes} dans la partie C de l'individu par rapport au nombre maximal de termes possibles *MAXTERMS*. La probabilité est définie par : $\frac{0.9\times(Nb_{termes}-1)}{MAXTERMS-1}$

Opérateur de spécialisation
L'opérateur de spécialisation rajoute un terme choisi aléatoirement à la règle. Nous avons testé notre algorithme avec et sans cet opérateur et avons observé que celui-ci n'apporte pas d'amélioration dans le temps de recherche et la qualité des solutions trouvées. Nous avons donc décidé de ne pas l'utiliser.

6.4.5 Sélection et insertion

L'objectif de l'algorithme est de trouver plusieurs règles différentes et il est donc nécessaire d'éviter une convergence de la population vers une seule règle.
Nous avons donc mis en place un mécanisme d'insertion dans la population qui empêche les doublons : on ne peut pas insérer un individu qui est déjà présent dans la population.
La sélection d'un individu se fait proportionnellement à sa qualité normalisée par rapport au meilleur individu f_i/f_{best} (sélection par le rang [Bak85]). La probabilité de choisir un individu de rang r parmi P individus de la population est donnée par la formule suivante :

$$p_r = \frac{1}{P(1+s)}\left(\frac{r}{P-1}+s\right)$$

où s correspond à la pression de sélection.
Ainsi les individus de bonne qualité ont un taux de reproduction plus important.

6.4.6 Choix adaptatif de mutation

Nous avons deux opérateurs de mutation : la mutation par valeur et la mutation par attribut. Nous reprenons les principes énoncés en 4.4.2.3.1 dans un cadre plus classique. Nous rappellons ici les notations. Dans [HWC00], les auteurs proposent de calculer ce nouveau taux de mutation en évaluant le progrès d'une mutation M_i pour un individu ind muté en un individu mut :

$$progress(M_i) = Max(fitness(ind), fitness(mut)) - fitness(ind)$$

Ensuite pour toutes les mutations, on calcule le ratio des gains relativement à $Nb\_mut(M_i)$ le nombre de mutations effectuées par l'opérateur M_i :

$$Gain(M_i) = \frac{progress(M_i)/Nb\_mut(M_i)}{\sum_j \left(progress(M_j)/Nb\_mut(M_j)\right)}$$

On fixe un taux de mutation minimum δ et un taux de mutation global $p_{mutation}$ pour N opérateurs de mutation à appliquer. On obtient pour le calcul des nouveaux taux de mutation :

$$p(M_i) = Gain(M_i) \times (p_{mutation} - N \times \delta) + \delta$$

La somme des taux de mutation est égale au taux de mutation $p_{mutation}$. Le taux de mutation initial est fixé à $p_{mutation}/N$.

6.5 Évaluation de l'approche

Dans le paragraphe 6.4, nous avons présenté la structure générale de l'algorithme génétique ASGARD. Nous présentons maintenant une évaluation de cet algorithme par son application sur une base de données classique pour pouvoir ensuite l'appliquer à des bases de données génomiques. Nous avons tout d'abord expérimenté cet algorithme sur la base de données *Nursery School* [BM98]. Cette base publique a été utilisée par différents auteurs afin d'évaluer leurs approches [NFL99, ALF99]. Nous connaissons donc certains résultats et pouvons nous comparer aux meilleurs résultats de la littérature.
La base *Nursery School* permet la classification de candidats à une école maternelle. Elle est formée de 12 960 instances et de 9 attributs (voir tableau 6.3). La décision finale pour accepter ou rejeter un candidat est prise en fonction de trois critères :

- l'emploi des parents,
- la situation financière et structurelle de la famille,
- la situation sociale et médicale de la famille.

L'attribut but choisi est *recommendation* et il peut prendre cinq valeurs : not_recom, recommend, very_recom, priority, spec_prior.

6.5.1 Expérimentation

Les tests ont été effectués sur un PC 600Mhz sous Linux. Le code a été réalisé en C. Les paramètres de l'algorithme génétique sont :

- Taille de la population : 100
- Nombre de générations : 100
- $P_{mutation}$: 0.9

Afin de pouvoir se comparer aux résultats existants, l'algorithme a été utilisé dans un contexte d'apprentissage en spécifiant certains buts à prédire ainsi l'expérimentation a été réalisée pour la recherche de règles ayant comme attribut but $recommendation$. Le tableau 6.4 indique les différentes valeurs de l'attribut *Recommendation*. Les valeurs $recommend$ et $very\_recom$ étant marginales, nous nous sommes intéressés aux trois valeurs les plus fréquentes.

	Nom de l'attribut	Valeurs de l'attribut
1	parents	usual, pretentious, great_pret
2	has_nurs	proper, less_proper, improper, critical, very_crit
3	form	complete, completed, incomplete, foster
4	children	1, 2, 3, more
5	housing	convenient, less_conv, critical
6	finance	convenient, inconv
7	social	nonprob, slightly_prob, problematic
8	health	recommended, priority, not_recom
9	recommendation	not_recom, recommend, very_recom, priority, spec_prior

TAB. 6.3 – Structure de la base de données *Nursery School*.

Valeur	Fréquence	%
not_recom	4320	33,33%
recommend	2	0,02%
very_recom	328	2,53%
priority	4266	32,92%
spec_prior	4044	31,20%

TAB. 6.4 – Structure de la base de données *Nursery School* pour l'attribut Recommendation.

6.5.2 Résultats

Qualité des règles

Nous donnons dans le tableau 6.5 les trois meilleures règles (par rapport à la mesure $J1$) obtenues pour chacune des trois valeurs de plus forte représentation pour l'attribut but Recommendation du tableau 6.4. Nous indiquons pour chaque règle la valeur de la mesure $J1$, la valeur de la couverture $\frac{|C|}{N}$, de la complétude $\frac{|C\&P|}{P}$, de la confiance $\frac{|C\&P|}{C}$, ainsi que le nombre minimum d'évaluations réalisées avant d'obtenir la règle.

En grisé sont indiquées les meilleures règles obtenues par Araujo et Freitas [ALF99] qui ont développé un algorithme génétique visant à optimiser la mesure $J1$. Si l'on considère le critère de la mesure $J1$, nous constatons que :

- Pour le but recommendation=priority, nous obtenons la meilleure règle de Freitas ainsi que d'autres règles de bonne qualité.
- Pour le but recommendation=not_recom, nous obtenons de meilleures règles que Araujo et Freitas qui ne présentent que des règles avec des mesures $J1$ quasi nulles, de très faible couverture (Couv=0.056) et de faible confiance (Conf=0.33) alors que notre algorithme trouve des règles de couverture de 0.333 et de confiance de 1.
- Pour le but recommendation=spec_prior, nous obtenons la meilleure règle de Freitas ($J1 = 0.033$) mais également des règles de meilleure qualité ($J1 = 0.0499$).

Ainsi, pour le critère de la mesure $J1$, l'algorithme proposé apparaît performant sur ce type de

J1	**Règles découvertes** Condition	**Couv.** $\frac{\|C\|}{N}$	**Compl.** $\frac{\|C\&P\|}{P}$	**Conf.** $\frac{\|C\&P\|}{C}$	**Nb_Eval**
But : recommendation=priority					
0.0427	IF ((health=recommended)) THEN (But)	0,333	0,566	0,558	806
0,0230	IF ((health=priority) AND (parents=usual)) THEN (But)	0,11	0,226	0,670	478
0,0225	IF ((health=recommended) AND (parents=usual)) THEN (But)	0,11	0,224	0,665	155
But : recommendation=not_recom					
0,159	IF ((health=not_recom)) THEN (But)	0,333	1,00	1,00	803
0,079	IF ((fi nance=inconv) AND (health=not_recom)) THEN (But)	0,16	0,50	1,00	0
0,0530	IF ((health=not_recom) AND (parents=usual)) THEN (But)	0,11	0,333	1,00	400
⋮	⋮	⋮	⋮	⋮	⋮
0,0000	IF housing=convenient AND fi nance=inconv THEN (But)	0,056	0,333	0,33	Non sélect. par ASGARD
But : recommendation=spec_prior					
0,0499	IF ((health=priority)) THEN (But)	0,33	0,609	0,517	959
0,0417	IF ((health=priority) AND (parents=great_pret)) THEN (But)	0,11	0,304	0,856	0
0,0330	IF ((health=priority) AND (has_nurs=very_crit)) THEN (But)	0,066	0,065	0,989	194

TAB. 6.5 – Résultats obtenus sur la base de données *Nursery School* lors d'une exécution. Les lignes grisées sont les meilleures règles trouvées par Araujo et Freitas [ALF99].

base de données.
Ce tableau indique également la valeur d'autres mesures pour ces règles et montre qu'une règle très bonne par rapport à un critère, peut l'être beaucoup moins par rapport à un autre. Ce tableau montre l'importance fondamentale du choix du critère en fonction des règles recherchées. Pour valider

Règles	**Support (%)**	**Confiance (%)**
But : recommendation=priority		
IF health=recommended THEN (But)	18,6	55,8
IF health=priority AND parents=usual THEN (But)	7,5	67,1
IF health=recommended AND parents=usual THEN (But)	7,4	66,5
IF health=recommended AND finance=inconv THEN (But)	7,4	66,5
IF health=recommended AND social=slightly_prob THEN (But)	6,9	62,3
But : recommendation=not_recom		
IF health=not_recom THEN (But)	33,3	100
IF health=not_recom AND finance=inconv THEN (But)	16,7	100
IF health=not_recom AND finance=convenient THEN (But)	16,7	100
IF health=not_recom AND parents=usual THEN (But)	11,1	100
IF health=not_recom AND parents=great_pret THEN (But)	11,1	100
But : recommendation=spec_prior		
IF health=priority THEN (But)	19	51,7
IF health=priority AND finance=inconv THEN (But)	10,1	60,6
IF health=priority AND parents=great_pret THEN (But)	9,5	85,6
IF health=priority AND finance=convenient THEN (But)	8,9	53,6
IF parents=great_pret AND finance=inconv THEN (But)	8,4	50,6

TAB. 6.6 – Résultats avec la méthode Apriori.

nos résultats, nous avons également généré avec l'algorithme Apriori [AS94, AS95a, SA96] toutes les règles ayant pour but l'un des trois étudiés. Pour cela, nous avons dû intégré à Apriori la possibilité de générer des règles avec but fixé. Dan le cas de cette application, Apriori travaille sur une base de données de 31 "items" puisque chaque valeur d'attribut correspond à un attribut. La taille de la base de donnée est donc relativement faible et cela permet de générer tous les sous ensembles fréquent d'items en mettant un seuil pour le support très faible mais non nul. En effet, si le support est mis à zéro, des règles abérrantes vont faire leur apparition. Mais dans notre application à des bases de données génomiques nous verrons les limitations d'Apriori.

Le tableau 6.6 présente les cinq meilleures règles obtenues par Apriori. Ces règles sont classées par but (comme pour le tableau 6.5) et ordonnées par support décroissant. En effet, le support est le critère principal de l'algorithme Apriori. En grisé sont indiquées les règles présentées au tableau 6.5, lorsqu'elles font parties des cinq meilleures.
Nous remarquons que les meilleures règles trouvées par l'algorithme génétique se confondent avec les meilleures règles d'Apriori ce qui confirme l'intérêt de l'utilisation des algorithmes génétiques. Notons que la troisième règle trouvée par l'algorithme génétique pour le but *recommendation = spec_prior* ne fait pas partie des meilleures règles d'Apriori. Cela s'explique par le support relativement faible de la règle (support=6.6%). Or cette règle a une confiance très élevée (Conf=99%), et est donc intéressante. L'algorithme génétique permet donc d'obtenir également des règles rares mais significatives, ce qui n'est pas le cas de l'algorithme Apriori qui élimine ces règles par l'utilisation du support.

Parcours de l'espace de recherche
L'espace de recherche global sur cette base de données est calculé en considérant que l'on peut prendre et ne pas prendre un attribut, et que l'on peut choisir pour cet attribut une valeur parmi celles possibles. Cela correspond à un arrangement de une valeur parmi les p possibles soit $A_p^1 = \frac{p!}{p-1!}$. Pour *Nursery School* nous considérons que nous avons un espace de recherche de $A_4^1 \times A_6^1 \times A_5^1 \times A_5^1 \times A_4^1 \times A_4^1 \times A_4^1 \times A_4^1 \times A_5^1$ soit 768000 solutions. Le nombre d'évaluations nécessaires pour l'obtention des solutions (colonne 6 du tableau 6.5) nous montre que la taille de l'espace de recherche parcouru pour trouver ces règles est inférieure à 0,2% de la taille de l'espace total.

6.5.3 Stabilité de l'algorithme

Le tableau 6.7 donne les valeurs d'une analyse statistique descriptive faite sur les résultats obtenus après 25 exécutions pour chaque classe. L'analyse est réalisée classe par classe sur la fitness du meilleur individu obtenu et sur la moyenne des fitness de la population de n individus X_i. Cette analyse donne la valeur du minimum, du maximum, de la moyenne m, de la médiane, de l'erreur standard $\frac{\sigma^2}{\sqrt{n}}$ avec $\sigma^2 = \sum^n \frac{(X_i - m)^2}{n}$ et de l'écart type σ.

Cette analyse statistique nous montre que l'algorithme est relativement robuste. En effet :
- l'écart type est relativement faible : l'ensemble des solutions trouvées sont de qualités semblables d'une exécution à l'autre ;
- la valeur de la médiane du meilleur individu est égale au maximum : la meilleure solution a été trouvée pour au moins 50% des exécutions et 100% pour "Recommendation=spec_prior" puisque la valeur du minimum est égale à celle du maximum.

Adaptativité des mutations
La figure 6.6 montre un exemple d'évolution des probabilités de mutation pour chaque opérateur en fonction des générations. Il apparaît que, pour cette exécution, même si la mutation par attribut semble globalement plus intéressante, à certains moments de la recherche, la mutation par valeur apporte également des améliorations, d'où l'intérêt d'un ajustement adaptatif des probabilités de mutation.

But	Mesure	Meilleur Individu	Moyenne Population
Recommendation Priority	Min	$2{,}3216\ 10^{-2}$	$1{,}2134\ 10^{-2}$
	Max	$4{,}2708\ 10^{-2}$	$1{,}4711\ 10^{-2}$
	Moyenne	$4{,}1928\ 10^{-2}$	$1{,}3947\ 10^{-2}$
	Médiane	$4{,}2708\ 10^{-2}$	$1{,}397\ 10^{-2}$
	Erreur standard	$0{,}0779\ 10^{-2}$	$9{,}2241\ 10^{-5}$
	Écart type	$0{,}3898\ 10^{-2}$	$0{,}0461\ 10^{-2}$
Recommendation not_recom	Min	$2{,}6507\ 10^{-2}$	$1{,}1840\ 10^{-2}$
	Max	$15{,}904\ 10^{-2}$	$4{,}8242\ 10^{-2}$
	Moyenne	$12{,}9352\ 10^{-2}$	$3{,}1649\ 10^{-2}$
	Médiane	$15{,}904\ 10^{-2}$	$3{,}4574\ 10^{-2}$
	Erreur standard	$1{,}0072\ 10^{-2}$	$0{,}1745\ 10^{-2}$
	Écart type	$5{,}0362\ 10^{-2}$	$0{,}8727\ 10^{-2}$
Recommendation spec_recom	Min	$4{,}991\ 10^{-2}$	$1{,}665\ 10^{-2}$
	Max	$4{,}991\ 10^{-2}$	$1{,}8654\ 10^{-2}$
	Moyenne	$4{,}991\ 10^{-2}$	$1{,}7571\ 10^{-2}$
	Médiane	$4{,}991\ 10^{-2}$	$1{,}751\ 10^{-2}$
	Erreur standard	0	$7{,}8831\ 10^{-5}$
	Écart type	0	$0{,}0394\ 10^{-2}$

TAB. 6.7 – Analyse statistique descriptive des résultats obtenus sur 25 exécutions sur les données de l'UCI.

6.6 Un outil de visualisation de règle d'association

Il existe quelques outils de visualisation de règles d'association dans la littérature. Ces représentations ont pour but de faciliter à l'utilisateur l'exploitation du résultat donné par des algorithmes d'extraction de règle. Mais les principales visualisations sont seulement disponibles dans des logiciels de datamining commerciaux.
Les algorithmes de recherche de règles d'association ont tendance à générer beaucoup de règles. Pour permettre à l'utilisateur de visualiser les règles générées, de les comparer les unes par rapport aux autres, nous avons entrepris avec l'aide de deux étudiants de DESS IAGL de Lille, de proposer un outil de visualisation de règles d'association permettant de comparer les règles par rapport à plusieurs critères de qualité. Cet outil a été entièrement réalisé en Java pour des soucis de portabilité. L'interface 3D a été réalisée grâce à l'API JAVA3D.
Notre logiciel "Association Rules Viewer" (ARV) propose trois représentations. L'idée d'une telle application est de pouvoir comparer des règles obtenues selon les critères d'optimisation $J1$ mais également à d'autres mesures largement répandues : support (supp.), confiance (conf), complétude (compl.), ... Cette application est indépendante de l'algorithme d'extraction de règles et prend en entrée un fichier XML. Le logiciel est disponible sur le web à l'adresse :
http ://www.lifl.fr/∼jourdan/download /.

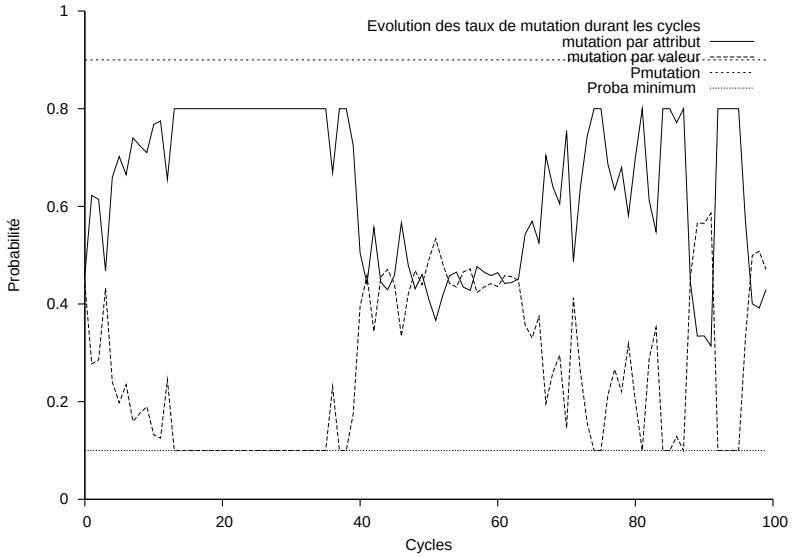


FIG. 6.6 – Evolution de la probabilité de mutation de chaque opérateur de l'algorithme génétique avec la mutation adaptative.

6.6.1 Visualisation 3D

Nous avons basé notre représentation 3D sur celle présentée dans [WWT99] qui peut visualiser beaucoup de règles d'association. Les lignes de la matrice deux dimension servant de support à la représentation représentent les attributs et les colonnes les associations d'attributs. Les blocs verts (resp. rouges) de chaque colonne (règle) représentent la condition (resp. ou l'antécédent) et la prévision (ou la conséquence). Les identités des attributs sont indiquées le long de la matrice.
Pour pouvoir évaluer chaque règle pas seulement grâce au support et à la confiance, nous ajoutons à la représentation 3D la possiblité de voir les différentes mesures de qualité calculées par ASGARD ou tout autre algorithme de génération de règles. De plus, pour plus de lisibilité, par rapport au travail de Wong [WWT99] qui a comme inconvénient une mauvaise lisibilité des deux critères (le premier cachant parfois le second), nous avons réalisé deux échelles différentes pour les critères. Le premier critère utilise les cinq graduations basses et le deuxième les cinq graduation hautes. Dans la représentation 3D, le bleu et le cyan représentent deux critères choisis. Notre système de visualisation comporte plusieurs fonctions de tri pour améliorer la lisibilité du résultat. On peut sur la visualisation 3D grâce à la souris réaliser des zoom, rotation et translation. La figure 6.7 montre une visualisation en trois dimensions des résultats obtenus sur la base de données *Nursery School*.

6.6.2 Visualisation de tous les critères

La deuxième visualisation est héritée de la visualisation N Dimensional Line [IB91] qui permet de comparer un grand nombre de règles par rapport à un grand nombre de critères. Chaque ligne représente une règle et chaque critère est un point de l'axe des abscisses. La valeur des critères est tracée sur l'axe des ordonnées. Nous avons pu remarquer qu'un des inconvénients de

cette visualisation vient de la perte de lisibilité lorsque plusieurs règles ont des valeurs proches et qu'elles se superposent. Pour remédier à ce problème, nous avons réalisé une normalisation de l'échelle afin de la rendre indépendante pour chaque critère. La figure 6.8 montre une visualisation de tous les critères des résultats obtenus sur la base de données *Nursery School*.

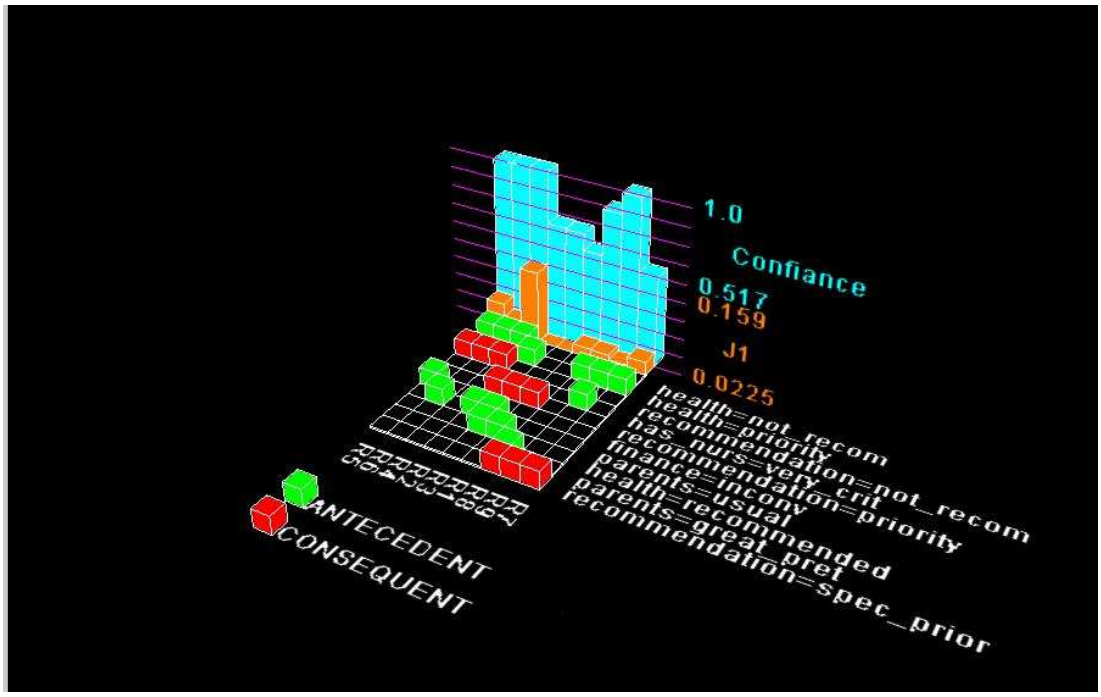


FIG. 6.7 – Les résultats obtenus par ASGARD sur le jeu de données Nursery visualisés par ARV en 3 Dimensions.

6.6.3 Double decker plot

Cette dernière visualisation [Unw00] diffère des précédentes dans la mesure où son objectif ne consiste pas à comparer plusieurs règles ensemble et suivant plusieurs critères. En effet l'objectif est d'attester de la qualité d'une règle en prenant toutes les valeurs possibles pour la partie gauche de la règle et en affichant les valeurs de confiance et de cardinalité. Une règle est jugée bonne si elle se distingue des autres par rapport à sa confiance.
Outre son caractère différent des autres car elle ne compare pas plusieurs règles d'associations ensemble, cette visualisation nécessite beaucoup de calculs. En effet, pour la visualisation d'une règle, il est nécessaire de calculer le nombre d'occurrences et la confiance pour chaque valeur possible de la condition et pour chaque règle que l'on voudra visualiser.

En appliquant ASGARD à une base de données classique, nous avons pu nous comparer à d'autres travaux et étudier la pertinence de cet algorithme (qualité des règles obtenues, efficacité de l'algorithme, intérêt du choix adaptatif de la mutation, stabilité). Notre objectif est maintenant d'appliquer cette méthode à une problématique réelle issue de la génomique.

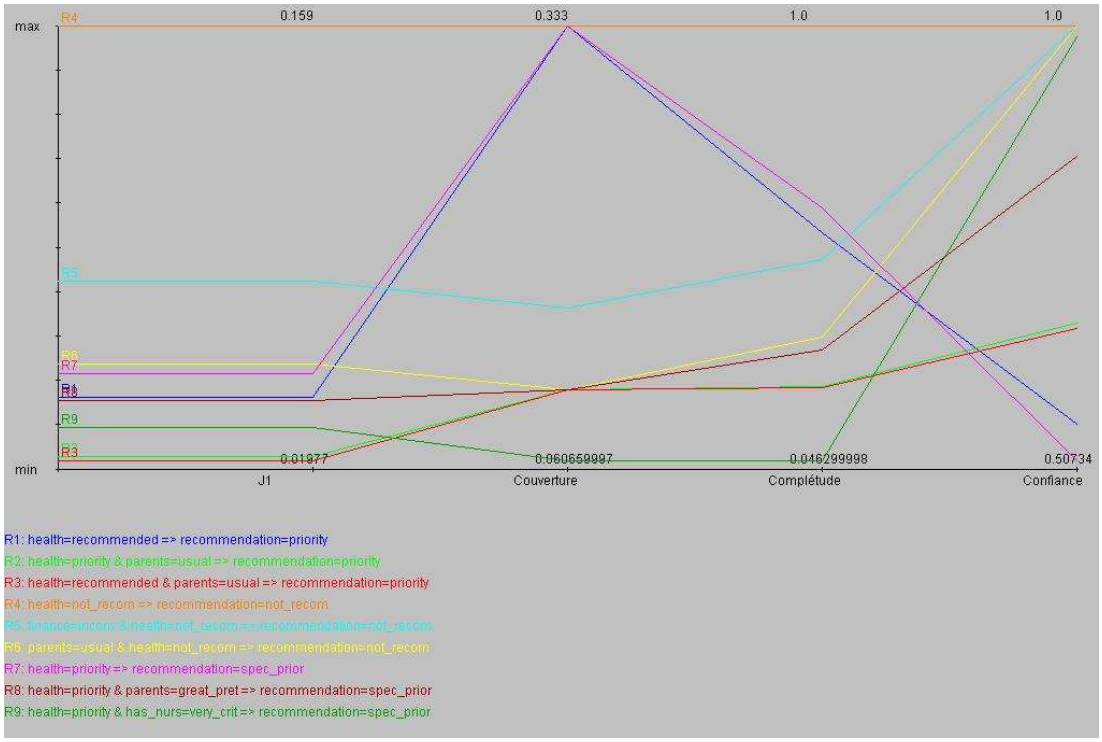


FIG. 6.8 – Les résultats obtenus par ASGARD sur le jeu de données Nursery visualisés par ARV en 2 Dimensions.

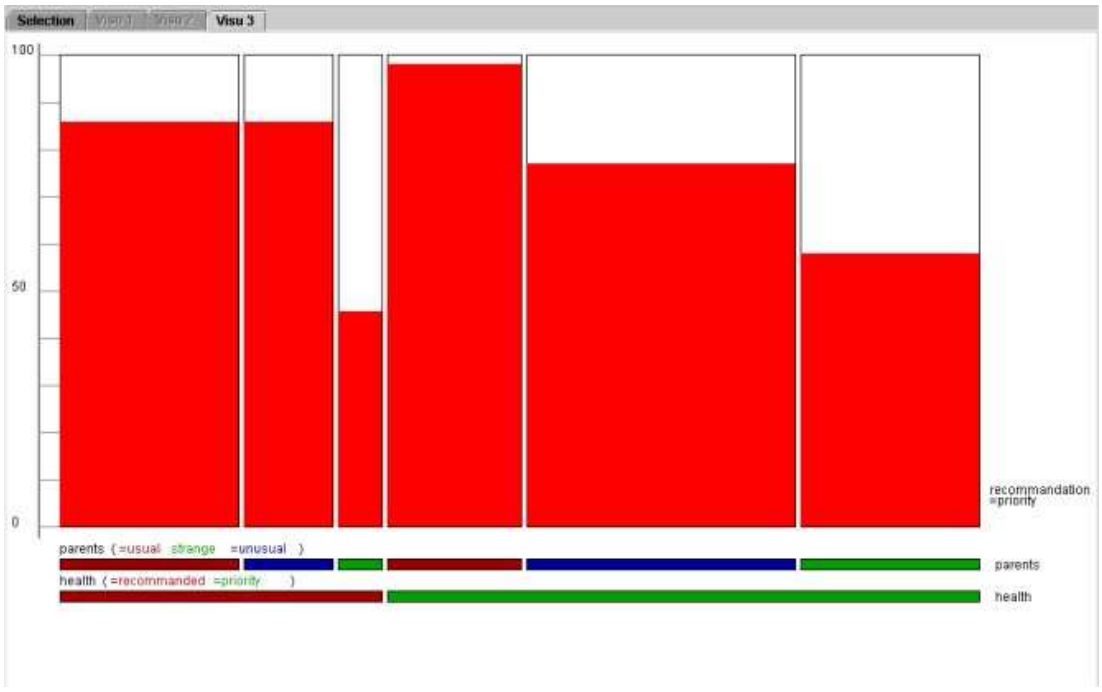


FIG. 6.9 – Exemple de double decker plot sur le jeu de données Nursery obtenu avec le logiciel ARV pour la règle "IF health=priority AND parents=usual THEN Recommendation=priority".

6.7 Application à la génomique

Cette étude porte sur une autre approche de l'étude du déséquilibre de liaison ("linkage desiquilibrium") pour les maladies multifactorielles et particulièrement sur l'étude d'associations de gènes candidats impliqués dans ces maladies. L'objectif est de rechercher des configurations de variants d'ADN (marqueurs) fréquents chez les malades et non fréquents chez les témoins. Nous avons détaillé dans le chapitre 3 la problématique biologique.
Notre objectif est de traiter ce problème biologique comme un problème de recherche de règles d'association et donc de ne pas utiliser les critères d'évaluation des biologistes mais de rechercher des associations entre les haplotypes, leur forme et la maladie. Nous nous positionnons donc dans le cadre particulier de recherche de règles de classification. Nous présenterons les adaptations nécessaires à l'algorithme présenté dans la première partie pour pouvoir traiter le problème en tenant compte des contraintes biologiques.

6.7.1 Adaptation et expérimentation

Nous avons tout d'abord adapté l'algorithme génétique pour qu'il prenne en compte les particularités du problème : déséquilibre de liaison, fréquence et valeurs manquantes.
Déséquilibre de liaison et fréquence :
Il faut en effet garantir la contrainte biologique que tous les SNPs composant un haplotype soient en déséquilibre entre eux. Pour cela, il faut vérifier que :

- leurs déséquilibres deux à deux soient inférieurs à un seuil S_1,
- leurs différences de fréquences soient supérieures à un seuil S_2.

Ces deux seuils seront fixés par les biologistes. Pour ne pas être trop sélectifs sur le déséquilibre, nous relâchons légèrement la contrainte donnée précédemment en prenant un nombre de déséquilibres pour l'association devant être supérieur à une borne minimale B_{min} calculée en fonction du nombre t de SNPs composant la partie C de la règle.

$$B_{min} = \frac{t \times (t-1)}{3}$$

Valeurs manquantes :
La base de données que nous traitons possède des valeurs manquantes. Ces valeurs manquantes ne sont pas intégrées au domaine possible des attributs afin de ne pas générer de règles avec des valeurs inconnues. De plus, nous avons intégré la gestion des valeurs manquantes pour le calcul de la fonction d'évaluation.

Ainsi, ne sont considérées dans l'évaluation d'une règle, que les instances pour lesquelles chaque attribut de la règle a une valeur connue. Le nombre total d'instances N peut donc être différent pour chacune des règles.

6.7.2 Résultats

Les tests ont été réalisés sur un PC 600Mhz sous Linux. Les seuils S_1 et S_2 ont été fixés par les biologistes respectivement à 0.4 et 0.05.

Les paramètres de l'algorithme génétique sont :

– Taille de la population : 100
– Nombre de générations : 150
– $P_{mutation}$: 0.9

L'espace de recherche pour la petite base biologique est $\left(A_4^1\right)^{51}$ puisqu'il y a 51 SNPs et qu'ils peuvent avoir 3 formes 11, 22 et 12 ou ne pas être choisis. L'espace de recherche est donc de l'ordre de 5.0706×10^{30}. Si on se rapporte à la formalisation du problème du chapitre 4, on peut restreindre l'espace de recherche à des solutions de taille 6 SNPs. La base de données de 51 SNPs n'est qu'un extrait des bases de données réellement utilisées qui possèdent 249 SNPs pouvant avoir trois formes.

J1	**Règles découvertes**	**Couv.** $\frac{\|C\|}{N}$	**Compl.** $\frac{\|C\&P\|}{P}$	**Conf.** $\frac{\|C\&P\|}{C}$	**Nb_Eval**	**Nb des**
0,0366	IF ((snp7=12) AND (snp16=11) AND (snp46=22)) THEN (aff=Atteint)	0,09	0,25	0,9167	2502	3
0,0343	IF ((snp7=12) AND (snp16=11) AND (snp36=11)) THEN (aff=Atteint)	0,189	0.3636	0,6400	2826	2
0,0291	IF ((snp6=11) AND (snp16=11) AND (snp46=22)) THEN (aff=Atteint)	0,25	0,409	0,5455	571	2
0,0120	IF ((snp21=11) AND (snp27=12) AND (snp29=11)) THEN (aff=Atteint)	0,136	0,2045	0,5000	282	2

TAB. 6.8 – Résultats obtenus sur la base de données biologique.

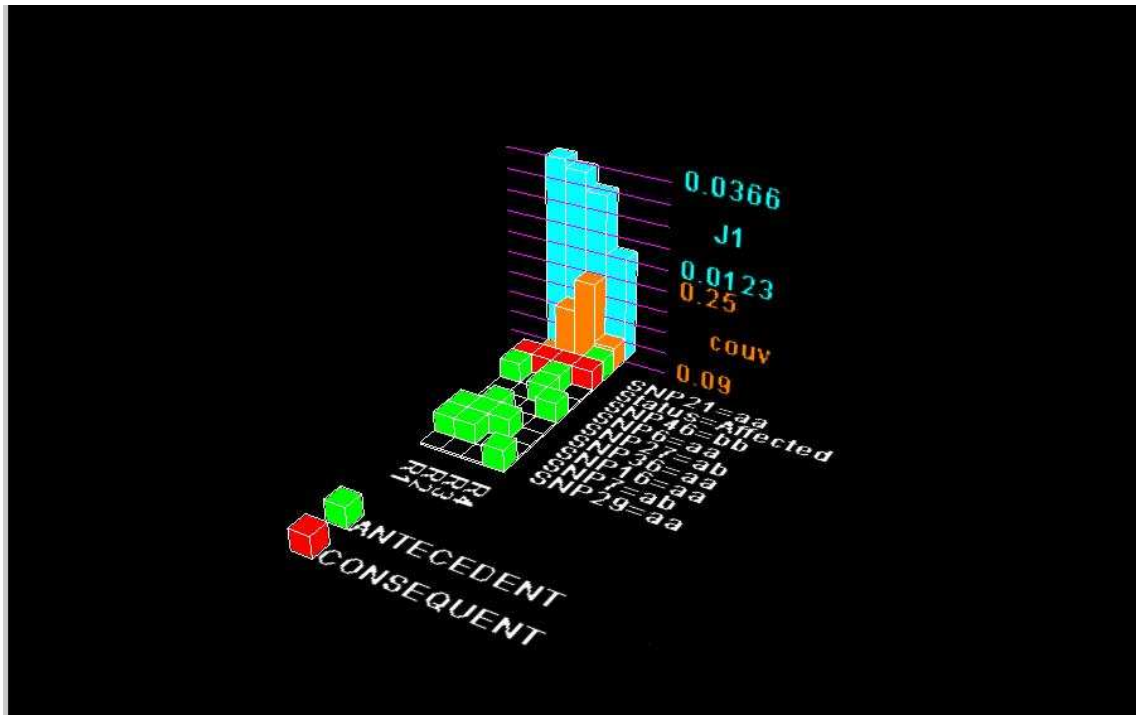


FIG. 6.10 – Les résulats obtenus par ASGARD sur le jeu de données biologiques visualisés par ARV en 3 Dimensions.

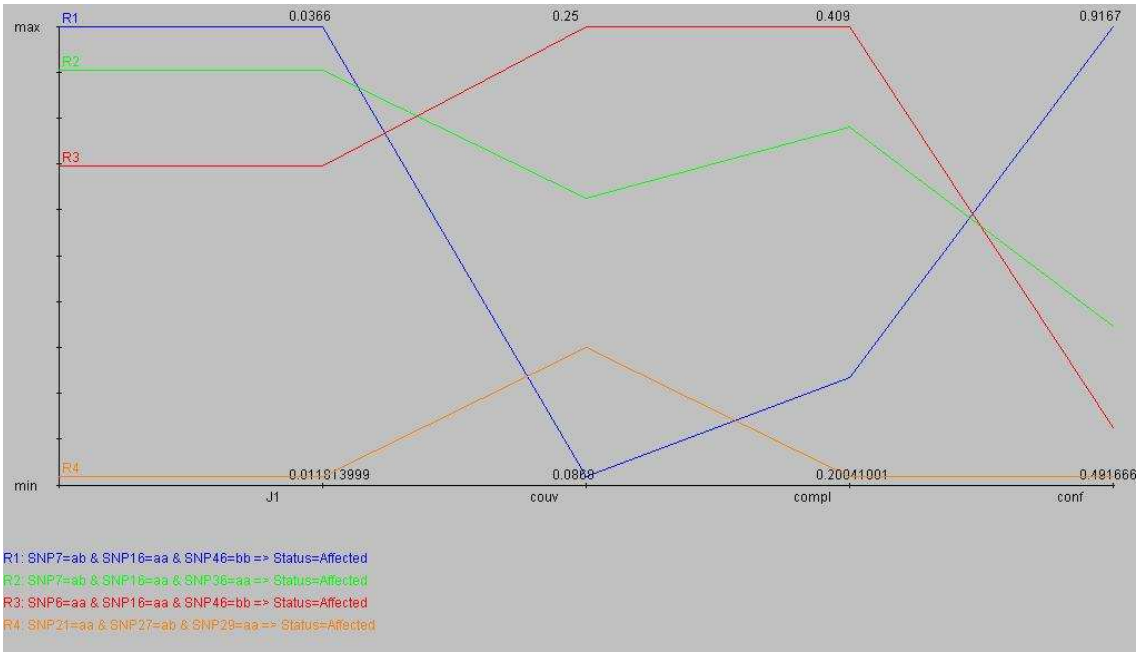


FIG. 6.11 – Les résulats obtenus par ASGARD sur le jeu de données biologiques visualisés par ARV en 2 Dimensions.

Mesure	Meilleur Individu	Moyenne Population
Min	$2{,}5812\ 10^{-2}$	$1{,}1328\ 10^{-2}$
Max	$3{,}6613\ 10^{-2}$	$2{,}1166\ 10^{-2}$
Moyenne	$3{,}2391\ 10^{-2}$	$1{,}9340\ 10^{-2}$
Médiane	$3{,}6613\ 10^{-2}$	$2{,}0897\ 10^{-2}$
Erreur standard	$0{,}1080110^{-2}$	$0{,}0171\ 10^{-2}$
Écart type	$0{,}3415\ 10^{-2}$	$0{,}054337\ 10^{-2}$

TAB. 6.9 – Analyse statistique descriptive des résultats obtenus sur 25 exécutions sur les données biologiques

Qualité des solutions

Le tableau 6.8 montre quelques règles obtenues lors d'une exécution de l'algorithme sur le jeu de données de 51 SNPs. Comme pour les données de l'UCI, nous indiquons les valeurs de la mesure $J1$, la couverture, la complétude, la confiance et le nombre d'évaluations. De plus, nous avons ajouté le nombre de déséquilibres de l'haplotype pour vérifier que la contrainte biologique est respectée. Nous remarquons que, malgré la taille de l'espace de recherche, il faut un nombre d'évaluations relativement faible pour obtenir une solution. Plusieurs haplotypes proposés comme solutions par l'algorithme génétique possèdent des attributs communs. Il existe en effet des SNPS fort présents dans différentes associations. Enfin, les résultats montrent qu'il serait intéressant d'approfondir l'étude des mesures permettant l'évaluation d'une règle car dans certains cas les règles ont des confiances élevées et de faibles couvertures.

La comparaison à Apriori dans ce cas est plus difficile. Tout d'abord, il faut intégrer à la génération de règles de la deuxième phase d'Apriori la possibilité de prendre en compte le déséquilibre de liaison. Ensuite, nous sommes vite arrivés aux limites de la version standard d'Apriori que nous possédions. En effet, les différents jeux de données que nous possédions ont un nombre relativement élevé d'attributs : 155 pour le jeu de données de 51 SNPs ($3 \times 51 + 2$ attributs de classes) et 749 ($3 \times 249 + 2$ attributs de classes) pour le jeux de données de 249 SNPs si l'on ne compte pas les valeurs manquantes. Lors de nos tests, sur le jeux de données de 51 SNPs, Apriori génère reltivement rapidement des règles de petites tailles en fixant un faible support. Mais lorsque l'on passe à la base de données de 249 SNPs, Apriori se révèle incapable de donner des résultats en un temps raisonnable (inférieur à vingt heures de calcul) lorsque nous descendons le support en dessous de 22%. Les limitations d'Apriori dans le version que nous possédons apparaissent à partir de 600 items de façon flagrante [Blo02].

Stabilité de l'algorithme

Nous avons réalisé plusieurs exécutions de l'algorithme ASGARD avec les mêmes paramètres. La meilleure solution obtenue est semblable d'une exécution à l'autre. Le tableau 6.9 montre les résultats d'une analyse statistique pour 25 exécutions de l'algorithme. Ce tableau montre que l'écart type sur la meilleure solution et sur la moyenne des solutions obtenues est relativement faible (respectivement $0.3415\ 10^{-2}$ et $0.054337\ 10^{-2}$). Nous pouvons remarquer que la dispersion des valeurs est très faible aussi bien sur le meilleur individu que sur la moyenne des individus, ASGARD montre donc une robustesse pour la convergence.

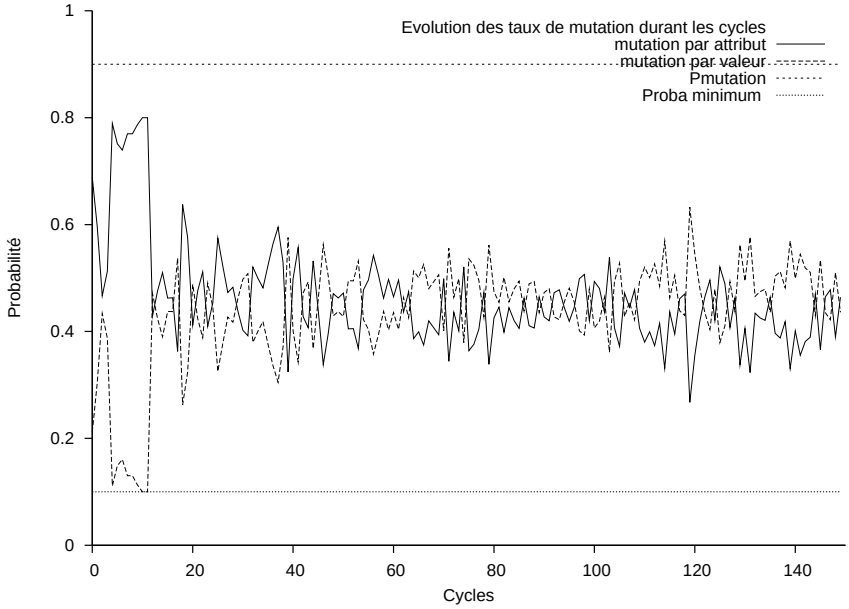


FIG. 6.12 – Evolution de la probabilité de mutation de chaque opérateur de l'algorithme génétique avec la mutation adaptative pour la problématique biologique

6.7.3 Adaptativité des mutations

La figure 6.12 présente les variations des probabilités de mutation des deux opérateurs de mutation : mutation par valeur, mutation par attribut. Il apparaît pour cette exécution que la mutation par attribut est plus performante au début de l'exécution alors que la tendance s'inverse pour être ensuite légèrement en faveur de la mutation par valeur. Nous pouvons enfin remarquer que l'allure de la figure 6.12 est très différente de celle obtenue pour la base de données "Nursery" de l'UCI (figure 6.6). Puisque d'une application à une autre, il est impossible de connaître a priori les performances des opérateurs de mutation, il est donc problématique d'imposer arbitrairement des taux de mutation et l'approche adaptative permet de contourner ce problème.

6.8 Conclusion

L'extraction de règles d'association peut donc se définir comme un problème d'optimisation. Nous avons proposé dans ce chapitre un algorithme génétique permettant de traiter des bases de données de type nominale. Cet algorithme, nommé ASGARD, possède un codage et des opérateurs adaptés à la recherche de règles d'association. Pour pouvoir aussi bien changer la taille des règles obtenues que les modifier, nous avons mis en place plusieurs mutations. Pour pouvoir appliquer selon l'évolution de l'algorithme ces différentes mutations et adapter leur taux d'application en fonction de l'amélioration des solutions qu'elles apportent, nous avons décidé de mettre en place un mécanisme adaptatif qui calcule d'une génération sur l'autre le nouveau taux d'application de chaque mutation. Cela permet de ne pas devoir les fixer arbitraiement ou expérimentalement. En

effet, l'évolution des taux de mutation nous a montré que d'un problème à l'autre, l'algorithme a un comportement différent. Cela renforce l'intérêt de l'adaptation des taux de mutation puisque, a priori, il n'est pas possible de savoir, pour tel ou tel problème, quels seront les meilleurs taux.
Nous remarquons que notre approche permet d'obtenir des règles intéressantes et de pallier à un défaut d'un Apriori classique par rapport aux règles rares car Apriori sélectionne dans un premier temps les règles par rapport au support puis seulement par rapport à la confiance. Nous avons également remarqué que pour des bases de données de relativement faible taille en génomique (249 SNPs), la version de base d'Apriori se révèle incapable de générer des règles de support inférieur à 22%.
Nous avons ensuite adapté cette approche pour notre problématique biologique sur l'étude des maladies multifactorielles par déséquilibre de liaison. Cette adaptation nous génère des règles répondant à des contraintes sur le déséquilibre de liaison et la fréquence des SNPs. Nous avons également rajouté un traitement spécifique pour les valeurs manquantes. L'algorithme a montré lors des différents tests une robustesse au niveau de la convergence ce qui permet son utilisation par les biologistes.
La modélisation du problème de règles d'association comme un problème d'optimisation montre l'importance des mesures de qualité d'une règle. Les mesures actuellement utilisées résultent fréquemment de l'agrégation de mesures simples parfois contradictoires de façon à évaluer une règle selon plusieurs critères. Les figures 6.10 et 6.11 montrent en effet pour des problématiques différentes, la dominance des règles quelques suivant certains critères. Une des perspectives de ce travail est de traiter le problème de règles d'association comme un problème multicritère afin de pouvoir combiner plus judicieusement différents critères. Cet aspect est actuellement étudié dans le cadre de la thèse de M. Khabzaoui.

Chapitre 7

Conclusion générale

Les travaux que nous avons réalisés dans le cadre de cette thèse ont débuté par mon mémoire de DEA en 2000 [Jou00] alors que l'équipe entamait une réflexion sur l'adaptation de méthodes d'optimisation combinatoire à des problèmes d'extraction de connaissances et plus particulièrement pour des applications à des problématiques biologiques relatives à la génomique.

Nous avons travaillé avec l'Institut de Biologie de Lille sur deux problématiques relatives à l'étude des maladies multifactorielles comme le diabète et l'obésité. L'objectif était de comprendre les besoins des biologistes en terme d'aide à l'exploitation des données expérimentales et de les modéliser en des problèmes d'extraction de connaissances.

Dans un premier temps, il nous a fallu appréhender les problématiques biologiques et les formaliser comme un problème d'extraction de connaissances.

Dans un second temps, nous avons réalisé un état de l'art sur les différentes méthodes d'optimisation pour l'extraction de connaissances afin de déterminer les points importants d'un tel travail.

Nous avons décidé d'aborder ce travail à l'aide de métaheuristiques en insistant sur l'importance de la fonction d'évaluation, du codage d'une solution et des opérateurs à appliquer.

Nous avons donc proposé différentes approches par métaheuristiques s'appliquant à différentes tâches d'extraction de connaissances : sélection d'attributs, clustering et règles d'association. Pour chacun de ces problèmes nous avons réfléchi à une fonction d'évaluation adaptée à ce que l'on voulait obtenir comme résultat. Pour l'une des applications, la fonction d'évaluation nous était fournie par les biologistes et pour permettre une bonne modélisation du problème nous avons réalisé une étude de son comportement et du paysage qu'elle induisait.

Dans le cadre de notre premier travail sur la sélection d'attributs pour l'étude des paires de germains, le codage binaire que nous avons utilisé pouvait permettre l'utilisation d'opérateurs dits naturels. Nous avons montré qu'il était intéressant pour cette application d'avoir des opérateurs dédiés au problème en comparant les résultats obtenus en appliquant les opérateurs naturels et des opérateurs dédiés à la sélection d'attributs. En effet, nous avons observé que les opérateurs dédiés permettent un meilleur parcours de l'espace de recherche et une meilleure robustesse de l'algorithme.

Dans la majorité des algorithmes génétiques que nous avons développés, nous avons utilisé plusieurs opérateurs de croisement et de mutation. Pour permettre leur utilisation conjointe sans re-

faire des expérimentations à chaque nouveau jeu de données pour fixer leur taux d'application, nous avons développé une méthode qui fixe les taux d'application des opérateurs de manière adaptative. Nous avons démontré expérimentalement son intérêt en montrant que certains opérateurs étaient plus ou moins performants en début ou en fin d'exécution selon les jeux de données et donc qu'il est nécessaire de garder chacun d'entre eux. Une telle stratégie adaptative permet d'utiliser un pool d'opérateurs ayant montré leurs performances dans certaines situations sans préjuger de leurs performances sur de nouvelles applications.

Nous avons également proposé d'intégrer, à certains de nos algorithmes, différents modèles parallèles en fonction de l'objectif : améliorer la robustesse de l'approche ou réduire son temps d'exécution. Nous avons ainsi mis en place dans l'étude des paires de germains un modèle distribué pour permettre une meilleure robustesse de notre approche en utilisant les principes du modèle en île. Le parallélisme ainsi mis en œuvre a permis d'améliorer la moyenne des solutions obtenues. Nous avons également proposé d'utiliser dans l'étude du déséquilibre un modèle de parallélisme centralisé afin de réduire le temps du processus d'évaluation qui est une phase critique de cette approche. Ceci a permis de pouvoir notamment générer des solutions plus rapidement.

Dans le cadre de notre travail sur le clustering, nous avons défini une nouvelle notion de centre de cluster, le centre partiel, auquel nous avons associé une distance et montré ses propriétés. Nous avons également démontré que notre notion de centre répondait bien aux critères nécessaires et notamment que le centre partiel est à distance minimale de tous les points du cluster. Cette notion de centre nous a permis de mettre en place un algorithme génétique hybridé avec une itération de Kmeans adapté aussi bien aux données numériques que nominales. Notre algorithme s'est montré capable aussi bien sur des jeux de données numériques que nominaux de trouver le nombre de clusters équivalent au nombre de classes définies dans le jeu de données. A l'inverse, la majorité des algorithmes classiques de clustering nécessitent qu'on leur indique soit un nombre de clusters soit une indication sur un seuil de similarité minimum [JMF99b]. Nos expérimentations nous ont montré la robustesse de notre méthode sur divers jeux de données provenant de l'UCI et la qualité des solutions trouvées par rapport à celles découvertes par des méthodes classiques telles que Kmeans ou Single Link. La généralité de l'algorithme pouvant traiter des données numériques ou des données nominales ainsi que la non nécessité de lui préciser un nombre de clusters le rendent intéressant pour l'étude de données réelles.

Nous avons également abordé le problème de recherche de règles d'association comme un problème d'optimisation. Nous avons validé notre approche dans un premier temps sur des jeux de données classiques de la littérature en nous comparant aux résultats d'autres travaux [ALF99, NFL99]. Ensuite, nous l'avons adaptée, en ajoutant des connaissances du domaine, à la recherche par déséquilibre de liaisons de formes de SNPs impliquées dans des maladies multifactorielles. Lors de nos expérimentations sur les bases de données génomiques, nous avons remarqué que l'algorithme le plus classique de la littérature, Apriori, se révèle dans sa version de base incapable de générer des règles de support inférieur à 22%.

Ce travail de thèse se positionne comme une contribution au développement d'algorithmes génétiques pour des problèmes d'extraction de connaissances et à leur application à des problématiques biologiques réelles. Notre contribution se situe dans la modélisation des problèmes et l'étude des différents choix possibles des opérateurs des algorithmes génétiques et dans leur adaptation pour pouvoir traiter des données issues de la génomique en intégrant directement dans la méthode de résolution des connaissances du domaine.

Perspectives

Dans ce mémoire, nous avons présenté une démarche de modélisation et de résolution de tâches d'extraction de connaissances en problèmes d'optimisation combinatoire. Nous avons jugé intéressant d'utiliser des métaheuristiques et notamment des algorithmes génétiques pour les résoudre dû à des problèmes de non-monotonicité des fonctions d'évaluation utilisées et à leur capacité à fournir des résultats exploitables en cours de recherche. Dans le cadre de l'équipe OPAC, les travaux présentés dans cette thèse ont une continuité aussi bien dans les problématiques que dans les approches.

En effet, en ce qui concerne la modélisation de problèmes, nous travaillons actuellement en collaboration avec la société de biotechnologie IT-OMICS sur des données d'expressions issues de puces à ADN pour mettre en évidence de nouveaux marqueurs pour les maladies cardiovasculaires. Nous avons modélisé ce problème comme un problème de recherche de règles d'association multicritères, critères dont une étude plus approfondie a été effectuée dans le cadre de la thèse de M. Khabzaoui [KDNT03]. Cette approche est actuellement en cours de développement et est portée sur PARADISEO qui est une plateforme de développement libre (open source) pour l'élaboration de métaheuristiques parallèles et distribuées hybrides. PARADISEO est développée en partie dans notre équipe dans le cadre de la thèse de S. Cahon [CTM03].

Dans le cadre de notre travail sur le clustering, nous souhaitons réaliser une étude comparative des critères permettant de déterminer le nombre optimal de clusters. Le clustering étant comme nous l'avons remarqué au chapitre 5 naturellement multicritère et traité par une combinaison des critères qui mesurent par exemple la compacité du cluster ou l'éloignement des clusters entre eux, nous aimerions proposer une modélisation multicritère du problème et réaliser une étude approfondie des différents critères possibles.

Dans ce mémoire, nous avons mis en place une hybridation d'un algorithme génétique avec l'algorithme Kmeans. Nous souhaitons maintenant trouver des schémas plus généraux d'hybridation de méthodes exactes avec des métaheuristiques. Nous allons notamment travailler sur la possibilité d'hybrider les algorithmes génétiques et la méthode Apriori pour permettre une recherche de règles d'association plus efficace. Nous proposons notamment d'utiliser l'algorithme Apriori pour éliminer les règles fréquentes et de permettre ainsi à l'algorithme génétique de trouver les règles les plus fréquentes dans ce qui est rare.

Bibliographie

[AC02] D.K. Agrafiotis and W. Cedeno. Feature selection for structure-activity correlation using binary particle swarms. *Journal of Medicinal Chemistry*, 45(5) :1098–1107, 2002.

[AD91] H. Almuallim and T. Dietterich. Learning with many irrelevant features. In AAAI Press, editor, *In Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547– 552, 1991.

[AD94] H. Almuallim and T. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2) :279–305, 1994.

[AIP01] A. Angiulli, G. Ianni, and L. Palopoli. On the complexity of mining association rules. In *Atti del Nono Convegno su Sistemi Evoluti per Basi di Dati (SEBD)*, page 8, Venise, Italie., 2001.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, May 1993. Washington DC, USA.

[ALF99] D.L.A. Araujo, H.S. Lopes, and A.A. Freitas. A Parallel Genetic Algorithm for Rule Discovery in Large Databases. In K. Ilto, editor, *Proc 1000 IEEE Systems, Man and Cybernetics Conf*, volume III, pages 940–945, Tokyo, October 1999. IEEE.

[Ali] Pr. Alicia. Cours VII : Déséquilibre de liaison. Cours en ligne. http ://anthro.unige.ch/GMDP/Alicia/GMDP_LD.htm.

[And73] M. R. Anderberg. *Cluster Analysis For Applications*. Academic Press, New York, 1973.

[And94] D. Andre. Learning and upgrading rules for an OCR system using genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, pages 462–467. IEEE Press, 1994.

[And97] D. Andre. Learning and upgrading rules for an optical character recognition system using genetic programming. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages G8.1 :1–8. Institute of Physics Publishing and Oxford University Press, Bristol, New York, 1997.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[AS95a] R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[AS95b] K. Al-Sultan. A tabu search approach to the clustering problem. *Pattern Recognition*, 28(9) :1443–1451, 1995.

[AS96] R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Trans. On Knowledge And Data Engineering*, 8 :962–969, 1996.

[Aug00] S. Augier. *Apprentissage Supervisé relationnel par Algorithmes d'évolution*. PhD thesis, Université Paris Sud, Orsay, France, 2000.

[AV96] S. Augier and G. Venturini. SIAO1 : A first order logic machine learning system using genetic algorithms. In *13th Int. Conf. on Machine Learning (ICML'96) – Workshop on Evolutionary Algorithms and Machine Learning, Bari, Italy*, July 1996.

[Azé03] J. Azé. Une nouvelle mesure de qualité pour l'extraction de pépites de connaissances. In *EGC 2003*, volume 17 of *RIA-ECA*, pages 171–182, 2003.

[BAB+96] C. Beyeler, M. Armstrong, H.A. Bird, J.R. Idle, and A.K. Daly. The role of cytochrome P450 2D6 in ankylosing spondylitis. *Ann Rheum Dis*, 55(1) :66–80, 1996.

[Bac99] V. Bachelet. *Métaheuristiques parallèles hybrides : Application au QAP*. PhD thesis, USTL LIFL France, 1999.

[Bak85] J.E. Baker. Adaptive selection methods for genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 101–111, 1985. Carnegie-Mellon University, Pittsburgh, USA.

[Bal94] S. Baluja. Population-based incremental learning : A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[Bar89] A. L. Barker. Neural network applications to sensor data fusion. Master's thesis, University of Virginia, USA, 1989.

[Bay98] R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 85–93, 1998. Whashington, USA.

[BB95] L. Bottou and Y. Bengio. Convergence properties of the K-means algorithms. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 585–592. The MIT Press, 1995.

[BBHB94] J.C. Bezdeck, S. Boggavaparu, L.O. Hall, and A. Bensaid. Genetic algorithm guided clustering. In *Proc. of the First IEEE Conference on Evolutionary Computation*, pages 34–38, 1994.

[BC92] G. Bianchi and R. Church. A non-binary encoded genetic algorithm for a facility location problem. Department of Geography, University of California, Santa Barbara, USA., 1992.

[BC95] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russel, editors, *The Int. Conf. on Machine Learning 1995*, pages 38–46, San Mateo, CA, 1995. Morgan Kaufmann Publishers.

[BDH+96] J. Bala, K. DeJong, J. Huang, H. Vafaie, and Harry Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, 4(3) :297–311, 1996.

[Bez98] J.C. Bezdek. Some new indexes of cluster validity. *IEEE Trans. on Systems, Man, and Cybernetics*, 28(3) :301–315, 1998.

[BFM00] T. Back, D. Fogel, and Z. Michalewicz. *Evolutionary Computation 1 : basic algorithms and operators*. Institute of Physic, 2000.

[BH65] G. Ball and D. Hall. Isodata, a novel method of data analysis and classification. Technical Report Technical Report AD-699616, SRI, Stanford, USA, 1965.

[BH90] D. E. Brown and C. L. Huntley. A practical application of simulated annealing to clustering. Technical Report IPC-91-03, Institute for Parallel Computation, University of Virginia, USA, 1990.

[BH92] D. E. Brown and C. L. Huntley. A practical application of simulated annealing to clustering. *Pattern Recognition*, 25 :401–412, 1992.

[BHV+95] J. Bala, J. Huang, H. Vafaie, K. DeJong, and H.Wechsler. Hybrid learning using genetic algorithms and decision tress for pattern classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 719–724, Montreal, Canada, 1995.

[BK93] K. D. Boese and A. B. Kahng. Simulated annealing of neural networks : the 'cooling' strategy reconsidered. In *IEEE Int. Symp. on Circuits and Systems*, pages 2572–2575, May 1993.

[BLF99] C. Bojarczuk, H. Lopes, and A.A. Freitas. Discovering comprehensible classification rules using genetic programming : a case study in a medical domain. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, pages 953–958, San Francisco, CA, 1999. Morgan Kaufmann.

[BLIS01] R. Blanco, P. Larrañaga, I. Inza, and B. Sierra. Selection of highly accurate genes for cancer classification by estimation of distribution algorithms. In *Workshop of Bayesian Models in Medicine*, pages 29–34, 2001.

[Blo02] F. Blondel. Etude d'apriori et application à la bioinformatique. Master's thesis, Eudil IMA, 2002.

[BM94] G. P. Babu and M. N. Murty. Connectionist approach for clustering. In *Proceedings International Conference on Neural Networks*, pages 4661–4666, 1994.

[BM98] C.L. Blake and C.J. Merz. Uci repository of machine learning databases. http ://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.

[BM01] S. Bandyopadhyay and U. Maulik. Nonparametric genetic clustering : Comparison of validity indices. *IEEE Trans. Syst., Man, Cybern-Part C : Applications and Reviews*, 2001.

[BM02] S. Bandyopadhyay and U. Maulik. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35 :1197–1208, 2002.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets : generalizing association rules to correlations. In *ACM SIGMAD*, pages 265–276, 1997.

[Bor01] R. Borg. Extracting fuzzy rules using genetic programming. Master's thesis, Delft University of Technology, Nederland, August 2001.

[BRLE98] D. Boichard, P. Le Roy, H. Levéziel, and J.-M. Elsen. Utilisation des marqueurs moléculaires en génétique animale. *INRA Production Animale*, pages 67–80, 1998.

[Car00] J.H. Carter. The immune system as a model for pattern recognition and classification. *Journal of the American Medical Informatics Association*, 7 :28–41, 2000.

[CB01] L.R. Cardon and J. Bell. Association study designs for complex diseases. *Nature Reviews Genetics*, pages 91–99, 2001.

[CDG+97] J. Chattratichat, J. Darlington, M. Ghanem, Y. Guo, and al. Large scale data mining : Challenges and responses. In D. Pregibon and R. Uthurusamy, editors, *Third International Conference on Knowledge Discovery and Data Mining*, pages 61–64. AAAI Press, 1997. Newport Beach, CA.

[CF01] D.R. Carvalho and A.A. Freitas. An immunological algorithm for discovering small-disjunct rules in data mining. In L. Spector, E.D. Goodman, A. Wu, W.B. Langdon, H-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, page 904, San Francisco, California, USA, 2001. Morgan Kaufmann.

[CGSS99] S. Chen, C. Guerra-Salcedo, and S.F. Smith. Non-standard crossover for a standard representation – commonality-based feature subset selection. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 129–134, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.

[CH74] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in statistics*, 3(1) :1–27, 1974.

[CHN+96] D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *PDIS : International Conference on Parallel and Distributed Information Systems*, pages 31–42. IEEE Computer Society Technical Committee on Data Engineering, and ACM SIGMOD, 1996.

[CL99a] K. Chen and H. Liu. Towards an evolutionary algorithm : A comparison of two feature selection algorithms. In *1999 Congress on Evolutionary Computation*, pages 1309–1313, Piscataway, NJ, 1999. IEEE Service Center.

[CL99b] K. Chen and H. Liu. Towards an evolutionary algorithm : A comparison of two feature selection algorithms. In *1999 Congress on Evolutionary Computation*, pages 1309–1313, Piscataway, NJ, 1999. IEEE Service Center.

[Col98] R.M. Cole. Clustering with genetic algorithms. Master's thesis, University of Western Australia, Australia, 1998. http ://citeseer.nj.nec.com/cole98clustering.html.

[Con95] C. B. Congdon. *A comparison of genetic algorithm and other machine learning systems on a complex classification task from common disease research*. PhD thesis, University of Michigan, 1995.

[CP02] E. Cantú-Paz. Feature subset slection by estimation of distribution algorithms. In *Gecco 2002*, pages 303–310. Margan Kaufman, 2002.

[CR00] S-C. Chu and J. F. Roddick. A clustering algorithm using the tabu search approach with simulated annealing. In *Data Mining II - Proc. Second International Conference on Data Mining Methods and Databases*, pages 515–523, 2000.

[CS96] K.J. Cherkauer and J.W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In AAAI Press, editor, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 315–318, Portland, OR, USA, 1996.

[CTM03] S. Cahon, E-G Talbi, and N. Melab. Paradiseo : A framework for parallel and distributed biologically inspired metaheuristic. In *BIOSP3 Workshop on Biologically Inspired Solutions to Parallel Processing Problems, IEEE IPDPS2003 (Int. Parallel and Distributed Processing Symposium)*, Nice, France, 2003.

[Dar59] C. Darwin. *On the Origin of Species*. John Murray, London, 1859.

[Dav89] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Third International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989. San Mateo, CA.

[DB79] D.L. Davies and D.W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1, 1979.

[dCZ00] L.N. de Castro and F.J. Von Zuben. An evolutionary immune network for data clustering. In *In Proceedings of the IEEE SBRN'00 (Brazilian Simposium on Artificial Neural Networks)*, pages 84–89, 2000.

[Deb00] K. Deb. *Evolutionary Computation 1 : basic algorithms and operators*, chapter Introduction to selection. Institute of Physic, 2000.

[DG89] J.L Deneubourg and S. Goss. Collective patterns and decision-making. *Ethology and Evolution*, pages 295–311, 1989.

[DL97] M. Dash and H. Liu. Feature selection for classification. *International Journal of Intelligent Data Analysis*, 1(3) :131–156, 1997.

[DL98] M. Dash and H. Liu. Hybrid search of feature subsets. In H.Y. Lee and H. Motoda, editors, *Proceedings of the Fifth Pacific Rim International Conference on AI : PRICAI'98*, pages 238–249, Singapore, 22–27 November 1998. Springer-Verlag.

[DL00] M. Dash and H. Liu. Feature selection for clustering. In Springer, editor, *PAKDD 2000, Kyoto, Japan*, pages 110 – 121, April 2000.

[dlIDRS96] B. de la Iglesia, Justin C. W. Debuse, and Victor J. Rayward-Smith. Discovering knowledge in commercial databases using modern heuristic techniques. In *Knowledge Discovery and Data Mining*, pages 44–49, 1996.

[dlIWRS+03] B. de la Iglesia, J. J. Wesselink, V. J. Rayward-Smith, J. L. Dicks, I. N. Roberts, V. Roberts, and T. Boekhout. *Metaheuristics : Computer Decision-Making, M.G.C*, chapter Developing Classification Techniques from Biological Databases using Simulated Annealing. Kluwer Academic Publishers, 2003.

[DLY97] M. Dash, H. Liu, and J. Yao. Dimensionality reduction for unsupervised data, 1997.

[DMC91] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical report, Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, italie, 1991.

[DMC96] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B : Cybernetics*, 26(1) :29–41, 1996.

[Dor92] M. Dorigo. *Learning and Natural Algorithms (in Italian)*. PhD thesis, DEI, Politecnico di Milano,Italy, 1992.

[DPV83] J.L. Deneubourg, J.M. Pasteels, and J.C. Verhaeghe. Probabilistic behaviour in ants : a strategy of errors ? *Journal of Theoretical Biology*, 105 :259–271, 1983.

[DR97] M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proc. 14th International Conference on Machine Learning*, pages 92–97. Morgan Kaufmann, 1997.

[DRR96] B. Devlin, N. Risch, and K. Roeder. Disequilibrium mapping : Composite likelihood for pairwise disequilibrium. *Genomics*, 36 :1–16, 1996.

[Dun73] J.C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact, well-seperated clusters. *Journal of Cybernetics*, 3(3) :32–57, 1973.

[EC00] V. Estivill-Castro. Hybrid genetic algorithms are better for spatial clustering. In *Pacific Rim International Conference on Artificial Intelligence*, pages 424–434, 2000.

[ECM97a] V. Estivill-Castro and A. Murray. Mining spatial data via clustering. Technical Report FIT-TR-1997-05, Callaghan 2308, Australia, November, 1997.

[ECM97b] V. Estivill-Castro and A. Murray. Spatial clustering for data mining with genetic algorithms. Technical Report FIT-TR-97-10, Callaghan 2308, Australia, 1997.

[ECM00] V. Estivill-Castro and A. M. Murray. Hybrid optimization for clustering in data mining. Technical Report 2000-01, Callaghan 2308, Australia, 2000.

[EHM00] C. Emmanouilidis, A. Hunter, and J. MacIntyre. A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *Congress on Evolutionary Computing 2000*, volume 2, pages 309–316. CEC, 2000.

[EHW86] A. El-Hamdouchi and P. Willett. Hierarchic document clustering using ward's method. In *SIGIR'86, Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, September 8-10, 1986*, pages 149–156. ACM, 1986.

[Eti01] J. Etienne. *Biochimie génétique, biologie moléculaire, 7e édition*. Masson, 2001. ISBN : 2294004477.

[Fal94] E. Falkenauer. A new representation and operators for genetic algorithms applied to grouping problems. *Evolutionary Computation*, 2(2) :123–144, 1994.

[Fal98] E. Falkenauer. *Genetic Algorithms and Grouping Problems*. John Wiley, 1998.

[FK00] P. Fränti and J. Kivijärvi. Randomised local search algorithm for the clustering problem. *Pattern Analysis & Application*, 3 :358–369, 2000.

[FLdA99] A.A. Freitas, H. Lopes, and D. Luis Alves de Araujo. A parallel genetic algorithm for rule discovery in large databases. In *Proceedings of Man and Cybernetics Conf. of Tokyo*, volume 3, pages 940–945. IEEE, October 1999.

[FLF00] M.V. Fidelis, H.S. Lopes, and A.A. Freitas. Discovery comprehensible classification rules with a genetic algorithm. In *CEC-2000*, pages 805–810, 2000.

[FM01] E. Falkenauer and A. Marchand. Using k-means ? consider arrayminer. In *Proceedings of the 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2001)*, June 2001.

[FM02] E. Falkenauer and A. Marchand. *Evolutionary Computation in Bioinformatics*, chapter Clustering Microarray Data with Evolutionary Algorithms, pages 219–230. Morgan Kaufman, 2002. ISBN 1-55860-797-8.

[FOW66] J. Fogel, J. Owens, and J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[FPAC94] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In IEEE Computer Society Press, editor, *In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212, Los Alamitos, CA, USA, 1994.

[FPS00] G. Folino, C. Pizzuti, and G. Spezzano. Genetic programming and simulated annealing : A hybrid method to evolve decision trees. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802, pages 294–303, Edinburgh, 2000. Springer-Verlag.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery*, chapter From Data Mining to Knowledge Discovery : An Overview., pages 1–34. MIT Press, 1996.

[Fre99] A.A. Freitas. On rule interestingness measures. *Knowledge-Based Systems Journal*, 12(5-6) :309–315, 1999.

[Fre01] A.A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery, 2001.

[Fre02] A.A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, 2002.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, 1990.

[Gar98] G. Gardarin. *Internet/Intranet et bases de données, Data Web, Data Media, Data Warehouse, Data Mining*. Eyrolles Informatique, 1998.

[GBD+94] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancbek, and V. Sunderam. *PVM : Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing,*. MIT PRess, 1994.

[GF96] A. Ghozeil and D. B. Fogel. Discovering patterns in spatial data using evolutionary programming. In MIT Press, editor, *Proceedings of the First Annual Conference Genetic Programming*, pages 521–527, July 1996.

[GH99] P. Galinier and JK. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3 :379–397, 1999.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability. The Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[Glo77] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1) :156 – 166, 1977.

[Glo86] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13 :533–549, 1986.

[GN95] A. Giordana and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3(4) :375–419, 1995.

[Gol89] D. E. Goldberg. *Genetic Algorithms - in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[GR87] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodalfunction optimization. In *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA)*, pages 41–49, Cambridge, MA, USA, 1987.

[Gra79] R. Gras. *Contribution à l'étude expérimentale et à l'analyse de certaines acquisitions scientifiques et de certains objectifs didactiques en mathématiques*. PhD thesis, Université Rennes I, France, 1979.

[Gre99] D.A. Greenberg. Summary of analyses of problem 2 simulated data for GAW11. *Genetic Epidemiology*, 17 :429–447, 1999. Suppl. 1.

[Gre00] J. Greffenstette. *Evolutionary Computation 1 : basic algorithms and operators*, chapter Rank-based selection. Institute of Physic, 2000.

[GS93] D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13(2/3) :229–257, Novembre/Décembre 1993. Edited by Grefenstette, J.J. Massachusetts : Kluwer Academic Publishers.

[GSCWS99] C. Guerra-Salcedo, S. Chen, D. Whitley, and S. Smith. Fast and accurate feature selection using hybrid genetic strategies. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 177–184, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[GSW98] C. Guerra-Salcedo and D. Whitley. Genetic search for feature subset selection : A comparison between CHC and GENESIS. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998 : Proceedings of the Third Annual Conference*, pages 504–509, University of Wisconsin, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

[GSW99] C. Guerra-Salcedo and D. Whitley. Genetic approach to feature selection for ensemble creation. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 236–243, Orlando, Florida, USA, 1999. Morgan Kaufmann.

[GT00] R. Gilleron and M. Tommasi. Découverte de connaissance à partir de données. Technical report, Grappa, Université Lille 3, France, 2000.

[Gud] D.F. Gudbjartsson. Multipoint linkage analysis based on allele sharing models. 2000.

[Gue03] A. Guenoche. Partitions optimisées selon différents critères : évaluation et comparaison. Technical report, Institut de Mathématique de Luminy, Marseille, 2003.

[Gui00] S. Guillaume. *Traitement des données volumineuses, mesures et algorithmes d'extraction de règles d'association et règles ordinales.* PhD thesis, Université de Nantes, France, Décembre 2000.

[Han86] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on Numerical Methods in Combinatorial Optimization*, 1986. Capri, Italy.

[Han93] S. Handley. Automatic learning of a detector for alpha-helices in protein sequences via genetic programming. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 271–278, University of Illinois at Urbana-Champaign, 1993. Morgan Kaufmann.

[HC01] J-P. Hugot and M. Chamaillard. Association of NOD2 leucine-rich repeat variants with susceptibility to Crohn's disease. *Nature*, 411 :599–603, 2001.

[HGD94] J. Horn, D.E. Goldberg, and K. Deb. Implicit niching in a learning classifier system : Nature's way. *Evolutionary Computation*, 1(2) :37–66, 1994.

[HKK97] E-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. In ACM, editor, *ACM SIGMOD Conference (Management of Data)*, volume 26, pages 277–288, 1997. New York.

[HKKM97] E-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *Research Issues on Data Mining and Knowledge Discovery*, 1997.

[HL96] F. Herrera and M. Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In F. Herrera and J. L. Verdegay, editors, *Genetic Algorithms and Soft Computing*, pages 95–125. Physica-Verlag, Heidelberg, 1996.

[HM02] J. Handl and B. Meyer. Improved ant-based clustering and sorting in a document retrieval interface. In J.J.M. Guervós, P. Adamidis, H-G. Beyer, J-L. Fernández-Villacañas, and H-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, pages 913–923, Berlin, 2002. Springer.

[HOB99] L. O. Hall, I. B. Oezyurt, and J. C. Bezdek. Clustering with a genetically optimized approach. *IEEE Transactions on evolutionary Computation*, 3(2) :103–112, July 1999.

[Hol75] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In W. Chen, J. Naughton, and P. A. Bernstein, editors, *2000 ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12. ACM Press, 05 2000.

[Hua98] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 1998.

[HWC00] T. P. Hong, H.S. Wang, and W.C. Chen. Simultaneosly applying multiple mutation operators in genetic algorithms. *Journal of Heuristics*, 6 :439 – 455, 2000.

[HWY+98] Z. He, C. Wei, L. Yang, X. Gao, S. Yao, R.C. Eberhart, and Y. Shi. Extracting rules from fuzzy neural network by particle swarm optimization. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pages 74–77, Anchorage, Alaska, USA, 1998.

[IB91] A. Inselberg and B.Dimsdale. Multidimensional lines I and II. *SIAM J. Appl. Math.*, 1991.

[ILS01a] I. Inza, P. Larrañaga, and B. Sierra., editors. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter Feature Subset Selection by Estimation of Distribution Algorithms (Chapter 13). Kluwer Academic Publishers, 2001.

[ILS01b] I. Inza, P. Larrañaga, and B. Sierra, editors. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter Feature Weighting for Nearest Neighbor by Estimation of Distribution Algorithms (Chapter 14). Kluwer Academic Publishers, 2001.

[Jan93] C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13(2/3) :189–228, 1993. Edited by Grefenstette, J.J. Massachusetts : Kluwer Academic Publishers.

[JB91a] D. R. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. In R.K Balew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 442–449. Morgan Kaufman Publishers, 1991.

[JB91b] D.R. Jones and M.A. Beltramo. Solving partitioning problems with genetic algorithms. In Eds. R. Belew, L. B. Booker, editor, *Proc. of the Fourth International Conference on Genetic Algorithms*, pages 442–449. Morgan Kaufman Publishers, 1991.

[JDT01a] L. Jourdan, C. Dhaenens, and E-G. Talbi. Algorithme génétique pour l'analyse des maladies multifactorielles. Journée Evolutionnaire Trimestrielle (JET'6), 2001.

[JDT01b] L. Jourdan, C. Dhaenens, and E-G. Talbi. A genetic algorithm for feature selection in data-mining for genetics. In *Metaheuristic International Conference 2001*, pages 29–34, 2001. Porto, Portugal.

[JDT+01c] L. Jourdan, C. Dhaenens, E-G. Talbi, S. Gallina, and C. Dina. A branch and bound procedure to analyse multifactorial diseases. In *European Mathematical Genetics Meeting*, pages 19–20, 2001.

[JDT02a] L. Jourdan, C. Dhaenens, and E-G. Talbi. ASGARD : un algorithme génétique pour les règles d'association. *Extraction de Connaissances et Apprentissage (ECA - Hermès)*, 16(6) :657–683, 2002.

[JDT02b] L. Jourdan, C. Dhaenens, and E-G. Talbi. *Evolutionary Computation in Bioinformatic.*, chapter Discovery of Genetic and Environmental Interactions in Disease Data using Evolutionary Computation, pages 297–316. Morgan Kaufmann, 2002.

[JDT03a] L. Jourdan, C. Dhaenens, and E-G. Talbi. Discovering haplotypes in linkage disequilibrium mapping with an adaptive genetic algorithm. In Günther R. Raidl, Stefano Cagnoni, Juan Jesús Romero Cardalda, David W. Corne, Jens Gottlieb, Agnès Guillot, Emma Hart, Colin G. Johnson, Elena Marchiori, Jean-Arcady Meyer, and Martin Middendorf, editors, *Applications of Evolutionary Computing, EvoWorkshops2003 : EvoBIO*, volume 2611 of *LNCS*, pages 66–75, University of Essex, England, UK, 2003. Springer-Verlag.

[JDT03b] L. Jourdan, C. Dhaenens, and E-G. Talbi. Rules extraction in linkage disequilibrium mapping with an adaptive genetic algorithm. In *European Conference on Computational Biology (ECCB) 2003*, pages 29–32, 2003. Paris, France.

[JDTG02] L. Jourdan, C. Dhaenens, E.G. Talbi, and S. Gallina. A data mining approach to discover genetic and environmental factors involved in multifactoral diseases. *Knowledge Based Systems*, 15(4) :235–242, May 2002.

[JMF99a] A.K. Jain, M.N. Murty, and P.J. Flynn. *Data Clustering : A Review*, volume 31, chapter 6, pages 263–323. ACM, Sept. 1999.

[JMF99b] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering : A review. *ACM Computing Surveys*, 31(3) :264–323, September 1999.

[Jou00] L. Jourdan. Data-mining pour la bio-informatique. Master's thesis, USTL Lille 1, 2000.

[JSG93] K. A. De Jong, W. M. Spears, and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13 :161–188, 1993.

[Jul95] B. A. Julstrom. What have you done for me lately ? adapting operator probabilities in a steady-state genetic algorithm. In L. J. Eshelman, editor, *Proceedings of the sixth International Conference on Genetic Algorithms*, pages 81–87. Morgan Kaufmann, 1995. San Francisco, CA.

[KD91] J D. Kelly and L. Davis. A hybrid genetic algorithm for classification. In *Proceedings of the 12th IJCAI*, pages 645–650. Morgan Kaufmann, 1991.

[KDNT03] M. Khabzaoui, C. Dhaenens, A. N'Guessan, and E-G. Talbi. Etude exploratoire des critères de qualité des règles d'association en datamining. In *XXXVèmes JOURNEES DE STATISTIQUE*, pages 583–586, 2003.

[KE95] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In IEEE Service Center, editor, *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948., 1995.

[Kep94] J.O. Kephart. A biologically inspired immune system for computers. In *Artificial Life IV : Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139, Cambridge, MA, US, 1994. MIT Press.

[KGV83] S. Kirkpatrick, D.C. Gelatt, and M.P. Vechhi. Optimization by simulated annealing. *Science*, 220 :671–680, May 1983.

[KJ97] R. Kohavi and G. John. Wrappers for feature subset selection. *AIJ Special Issue on relevance*, 97(1–2) :273–324, 1997.

[KJ98] R. Kohavi and G. John. The wrapper approach, 1998.

[KJDT03] M. Khabzaoui, L. Jourdan, C. Dhaenens, and E.G. Talbi. Approche évolutionnaire multicritère pour les règles d'association en génomique. In *Roadef 2003*, pages 173–174, 2003. Avignon, France.

[KL98] L. Kruglyak and E.S. Lander. Faster multipoint linkage analysis using fourier transform. *Journal of Computational Biology*, 5 :1–7, 1998.

[KM95] W.J. Krzanowski and F.H.C Marriot. *Multivariate Analysis*, chapter Cluster analysis, pages 61–94. London : Arnold, 1995.

[KM99] K. Krishna and M. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics - PartB : Cybernetics*, 29(3) :433—439, 1999.

[KN97] A. Kong and N.J.Cox. Allele sharing models : LOD scors and accurate linkage tests. *Americ Journal of Human Genetic*, 61 :1179–1188, 1997.

[Kod00] Y. Kodratoff. Quelques contraintes symboliques sur le numérique en ECD et en ECT, 2000.

[Koz92] J. R. Koza. *Genetic Programming : On the Programming of Computers by Natural Selection*. MIT Press, Cambridge, MA, 1992.

[KR92a] K. Kira and L. Rendell. The feature selection problem : traditional methods and a new algorithm. In AAAI Press/Cambridge, editor, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 129–134. MIT Press, 1992.

[KR92b] K. Kira and L. Rendell. A practical approach to feature selection. In Morgan Kaufmann., editor, *In Proceedings of the Ninth International Conference on Machine Learning.*, pages 249–256, 1992.

[KSM00] Y. S. Kim, W. N. Street, and F. Menczer. Feature Selection in Unsupervised Learning via Evolutionary Search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[LA03] I.C. Lerman and J. Azé. Une mesure probabiliste contextuelle discriminante de qualité des règles d'association. In *EGC 2003*, volume 17 of *RIA-ECA*, pages 247–262, 2003.

[Laz98] L. Lazzeroni. Linkage disequilibrium and gene mapping : an empirical least-squares approach. *American Journal of Human Genetics*, 62 :159–170, 1998.

[Ler98] Ph. Leray. *Apprentissage et Diagnostic de Systèmes Complexes : Réseaux de Neurones et Réseaux Bayésiens, Application à la gestion en temps réel du trafic téléphonique français*. PhD thesis, LIP6, 4, place Jussieu 75005 Paris (France), Septembre 1998.

[LF01] A. V. Lukashin and R. Fuchs. Analysis of temporal gene expression profiles : clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics*, 17(5) :405–414, 2001.

[LG87] E.S. Lander and P. Green. Construction of multilocus genetic linkage map in humans. *Proc. Natl. Acad. Sciences*, 84 :2363–2367, 1987.

[Lie83] M. Liebetrau. Measures of association. *Quantitative Applications in the Social Sciences Series*, 32, 1983.

[Liu68] G. L. Liu. *Introduction to combinatorial Mathematics*. McGraw Hill, 1968.

[LKM+99] P. Larraaga, C. Kuijpers, R. Murga, I. Inza, and S. Dizdarevich. Evolutionary algorithms for the travelling salesman problem : A review of representations and operators. *Artificial Intelligence Review*, 13 :129–170, 1999.

[LL99] J. A. Lozano and P. Larra. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20(9) :911–918, 1999.

[LL01] P. Larrañaga and J. Lozano, editors. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter A review on estimation of distribution algorithms (Chapter 3), pages 57–100. Kluwer Academic Publishers, 2001.

[LMP98] L. Lebart, A. Morineau, and M. Piron. *Statistique Exploratoire Multidimensionelle*. Dunod, 2 edition, 1998.

[LNY+00] W. Lee, R.A. Nimbalkar, K.K. Yee, S.B. Patil, P.H. Desai, T.T. Tran, and S.J. Stolfo. A data mining and CIDF based approach for detecting novel and distributed intrusions. *Lecture Notes in Computer Science*, 1907 :49–65, 2000.

[LS94] E.S. Lander and N.J. Schork. Genetic dissection of complex traits. *Science*, 265 :2037–2048, 1994.

[LS96] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In Morgan Kaufmann, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 284–292, 1996.

[LST+01] J.S. Liu., C. Sabatti, J. Teng, B. Keats., and N. Risch. Bayesian analysis of haplotypes for linkage disequilibrium mapping. *Genome Research*, 10(11) :1716–1724, 2001.

[LSW97] B. Lent, A. N. Swami, and J. Widom. Clustering association rules. In *Proc. of the ThirteenthInternational Conference of Data Engineering*, pages 220–231. IEEE, 1997.

[Maa] S. Maabout. *Règles d'association*. http ://www.labri.fr/Perso/m̃aabout/dess_gl/6asso_fr.ppt.

[Mas94] B. Masand. Optimising confidence of text classification by evolution of symbolic expressions. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 21. MIT Press, 1994.

[MB00] U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33 :1455–1465, 2000.

[MBP02] U. Maulik, S. Bandyopadhyay, and M.K. Pakhira. Clustering using annealing evolution : Application to pixel classifcation of satellite images. In *The III Indian conference on Computer Vision, Graphic and Image processing*, 2002. Oral session.

[MC85] G. Milligan and M. Cooper. An examination of procedures for detecting the number of clusters in a data set, 1985.

[Mer02] P. Merz. Clustering of gene expression profiles : An investigation of the fitness landscape of the minimum sum-of-squares clustering problem. In D. Corne, G. Fogel, W. Hart, J. Knowles, N. Krasnogor, R. Roy, J. Smith, and A. Tiwari, editors, *Advances in Nature-Inspired Computation : The PPSN VII Workshops*, pages 35–36, Reading, UK, 2002. PEDAL (Parallel, Emergent & Distributed Architectures Lab), University of Reading.

[Mic96] Z. Michalewizc. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, third, revised and extend edition, 1996.

[MN99] S. Morishita and A. Nakaya. Parallel branch-and-bound graph search for correlated association rules. In *Large-Scale Parallel Data Mining*, pages 127–144, 1999.

[Mon99] N. Monmarche. On data clustering with artificial ants. In A. Freitas, editor, *Data Mining with Evolutionary Algorithms : Research Directions*, pages 23–26, Orlando, Florida, 18 1999. AAAI Press.

[MP96] H. Mühlenbein and G. Paass. From recombination of genes to the estimation of distributions i. binary parameters. In *PPSN 1996*, pages 178–187, 1996.

[MSB01] H. Min, T. G. Smolinski, and G. M. Boratyn. A genetic algorithm-based data mining approach to profiling the adopters and non-adopters of e-purchasing. In W. W. Smari, editor, *Information Reuse and Integration, Third International Conference, IRI-2001*, pages 1–6. International Society for Computers and Their Applications (ISCA), 2001.

[MT01] N. Melab and E.-G. Talbi. A parallel genetic algorithm for rule mining. In *Workshop on Bio-Inspired Solutions to Parallel Processing Problems (BioSP3)*, page 133. José Rolim et al., Springer-Verlag, 2001. San Francisco, USA.

[MW90] O.L. Mangasarian and W.H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 1990.

[NF77] P.M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. on computer*, C-26(9) :917–922, September 1977.

[NFL99] E. Noda, A.A. Freitas, and H.S. Lopes. Discovery interesting prediction rules with a genetic algorithm. In *CEC-1999*, pages 1322–1329, 1999.

[NGCD02] O. Nasraoui, F. Gonzalez, C. Cardona, and D. Dasgupta. Artificial immune systems and data mining : Bridging the gap with scalability and improve learning. In *National Science Foundation Workshop on Next Generation Data Mining (NSF-NGDM)*, 2002.

[NGD02] O. Nasraoui, F. Gonzalez, and D. Dasgupta. The fuzzy artificial immune system : Motivations, basic concepts, and application to clustering and web profiling. In IEEE press, editor, *IEEE International Conference on Fuzzy Systems*, pages 711–716, 2002.

[OS94] D.W. Opitz and J.W. Shavlik. Using genetic search to refine knowledge-based neural networks. In Morgan Kaufman, editor, *Machine learning : Proceedings of the Eleventh International Conference*, pages 208–216, 1994.

[OS95] D. W. Opitz and J. W. Shalvik. *Computational Learning Thery and Natural Learning Systems*, chapter Vol. III, pages 3–19. MIT Press, 1995.

[OSE02] M. Omran, A. Salman, and A.P. Engelbrecht. Image classification using particle swarm optimization. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002)*, pages 370–374, 2002.

[Ove02] R. Overill. Computational immunology for fraud detection. Fourth Coordination Workshop and RETRIEVE end-of-project event, 2002.

[Pap76] C. H. Papadimitriou. *The complexity of combinatorial optimization problems*. PhD thesis, Princeton, 1976.

[PCY95] J. S. Park, M-S. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In M. J. Carey and D. A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 175–186, San Jose, California, 22–25 1995.

[PDPG93] M. Pei, Y. Ding, W. F. Punch, and E. D. Goodman. Classification and feature extraction of high-dimensionality binary patterns using GA to evolve rule. Technical report, GARAGe, 1993.

[Pet97] A. Petrowski. A new selection operator dedicated to speciation. In Thomas Bäck, editor, *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 144–151, San Francisco, 1997. Morgan Kaufmann.

[PGI97] M. Pei, E. D. Goodman, and W. F. Punch III. Pattern discovery from data using genetic algorithms. In *Proceedings of the First Pacic-Asia Conference on Knowledge Discovery and Data Mining*, February 1997. available via www URL : http ://garage.cps.msu.edu/papers/papers-index.html.

[PGL99] M. Pelikan, D. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. Technical Report Report No. 99018, IlliGAL, Illinois Genetic Algorithms Laboratory Department of General Engineering University of Illinois at Urbana-Champaign, USA, 1999.

[PGP97] M. Pei, M. Goodman, and W.F. Punch. Pattern discovery from data using genetic algorithm. In *Proc of the first Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Feb. 1997.

[PGPD95] M. Pei, E.D. Goodman, W.F. Punch, and Y. Ding. Genetic algorithms for classification and feature extraction. In *Annual Meeting : Classification Society of North America*, June 1995.

[PLF02] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas. *Data Mining : a Heuristic Approach*, chapter An Ant Colony Algorithm for Classification Rule Discovery, pages 191–208. London : Idea Group Publishing, 2002.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, 1982.

[PS91] G. Piatetski-Shapiro. Discovery, analysis, and presentation of strong rules. *Knowledge Discovery in Database*, 11 :229–248, 1991. AAAI/MIT Press.

[Rad91] N.J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5(2) :183–205, 1991.

[Rad92] N.J. Radcliffe. Genetic set recombinaison. In L.D. Whitley, editor, *Foundations of Genetic Algorithms*, volume 2, pages 203–219. Morgan Kaufmann, 1992. San Mateo, USA.

[Rec73] I. Rechenberg. *Evolutions Strategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.

[RFY01] B. J. Ross, F. Fueten, and D. Y. Yashkir. Automatic mineral identification using genetic programming. *Machine Vision and Applications*, 13(2) :61–69, 2001.

[RLS01] J. Roure, P. Larrañaga, and R. Sangüesa. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter An Empirical Comparison Between K-Means, GAs and EDAs in Partitional Clustering (Chapter 17). Kluwer Academic Publishers, 2001.

[RP99] A. Radi and R. Poli. Genetic programming discovers efficient learning rules for the hidden and output layers of feedforward neural networks. In R. Poli, P. Nordin, W. B. Langdon, and T. C. Fogarty, editors, *Genetic Programming : Second European Workshop EuroGP'99*, pages 120–134, Berlin, 1999. Springer.

[RPG+97] M.L. Raymer, W.F. Punch, E.D. Goodman, P.C. Sanschagrin, and L.A. Kuhn. Simultaneous feature extraction and selection using a masking genetic algorithm. Technical report, Michigan State University, USA, 1997.

[RR02] J.P. Reeve and B. Rannala. DMLE+ : Bayesian linkage disequilibrium gene mapping. *Bioinformatics*, 18(6) :894–895, 2002.

[RS03] A. Ratle and M. Sebag. A novel approach to machine discovery : Genetic programming and stochastic grammars. In S. Matwin and C. Sammut, editors, *ILP02*, volume 2583 of *LNAI*, pages 207–222. SV, 2003.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns : Generalizations and performance improvements. In P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 1996.

[SC95] P.C. Sham and D. Curtis. Monte carlo tests for associations between disease and alleles at hightly polymorphic loci. *Annal Human Genetics*, pages 97–105, 1995.

[Sch81] Hans-Paul Schwefel. *Numerical optimization of Computer models*. John Wiley & Sons, Ltd., Chichester, 1981.

[Sch01] T. Scheffer. Finding association rules that trade support optimally against confidence. In *Principles of Data Mining and Knowledge Discovery*, pages 424–435, 2001.

[SDB+03] Y. Saesys, S. Degroeve, B. De Baets, Y. Van de Peer, and P. Rouzé. feature selection using a simple estimation of distribution algorithm : A case study on splice site prediction. In *Belgian Bioinformatics Conference*, page 21, 2003.

[SG91] P. Smyth and R. M. Goodman. *Knowledge Discovery in Databases*, chapter Rule Induction Using Information Theory, pages 159–176. Piatetsky-Shapiro G. and Frawley J, 1991.

[SGW+95] R. Srikanth, R. George, N. Warsi, D. Prabhu, F.E. Petry, and B. P. Buckles. A variable-length genetic algorithm for clustering and classification. *Pattern Recognition Letters*, 16 :789–800, 1995.

[SHF00] A. Somayaji, S. Hofmeyr, and S. Forrest. Principles of a computer immune system. In *Meeting on New Security Paradigms, 23-26 Sept. 1997, Langdale, UK*, pages 75–82. New York, NY, USA : ACM, 1998, 2000.

[Sie94] E. V. Siegel. Competitively evolving decision trees against fixed training cases for natural language processing. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 19. MIT Press, 1994.

[SJI+01] B. Sierra, E. A. Jiménez, I. Inza, P. Larrañaga, and J. Muruzábal. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, chapter Rule Induction by Estimation of Distribution Algorithms (Chapter 15). Kluwer Academic Publishers, 2001.

[SK96] T. Shintani and M. Kitsuregawa. Hash based parallel algorithms for mining association rules. In *PDIS : International Conference on Parallel and Distributed Information Systems*, pages 19–30. IEEE Computer Society Technical Committee on Data Engineering, and ACM SIGMOD, 1996.

[SLI+01] B. Sierra, E. Lazkano, I. Inza, M. Merino, P. Larrañaga, and J. Quiroga. Prototype selection and feature subset selection by estimation of distribution algorithms. a case study in the survival of cirrhotic patients treated with tips. In *Proceedings of the 8th Artificial Intelligence in Medicine in Europe, AIME 2001*, pages 20–29, 2001.

[Smi80] S. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, USA, 1980.

[Smi83] S. Smith. Flexible learning of problem solving heuristics through adaptive search. In *Proceedings of the Eighth International Joint Conference on Articial Intelligence*,, pages 422–425, 1983. Karlsruhe, Germany.

[SNS03] T. Sousa, A. Neves, and A. Silva. Swarm optimisaton as a new tool for data mining. In *Proceedings of NIDICS*, Nice, France, 2003.

[SON95] A. Savasere, E. Omiecinski, and S. B. Navathe. An efficient algorithm for mining association rules in large databases. In *The VLDB*, pages 432–444, 1995. Zurich, Switzerland.

[SWBP01] B. A. Shapiro, J. C. Wu, D. Bengali, and M. J. Potts. The massively parallel genetic algorithm for rna folding : Mimd implementation and population variation. *Bioinformatics*, 17(2) :137–148, 2001.

[SZT02] I. Sarafis, Ams Zalzala, and P.W. Trinder. A genetic rule-based data clustering toolkit. In *Congress on Evolutionary Computation (CEC)*, 2002.

[Tac93] W. A. Tackett. Genetic generation of dendritic trees for image classification. In *Proceedings of the World Congress on Neural Networks*, pages IV646–IV649, 1993.

[Tal00] L. Talavera. "dependency-based feature selection for clustering symbolic data. *Intelligent Data Analysis*, 4 :19–28, 2000.

[TC02] J. Twycross and S. Cayzer. An immune-based approach to document classification. Technical Report HPL-2002-292, HP Laboratories, USA, 2002.

[Ter95] J.D. Terwilliger. A powerful likelihood method for the analysis of linkage disequilibrium between trait loci and one or more polymorphic marker loci. *American Journal of Human Genetics*, 56(3) :777–787, 1995.

[Tey01] O. Teytaud. *Théorie de l'apprentissage, Réseaux de neurones et applications.* PhD thesis, ERIC, Lyon 2 lumière, France, 2001.

[TK01] J. Timmis and T. Knight. Artificial immune systems : Using the immune system as inspiration for data mining. In Hussein A. Abbass, Ruhul A. Sarker, and Charles S.

Newton, editors, *Data Mining : A Heuristic Approach*, chapter XI, pages 209–230. Group Idea Publishing, September 2001.

[TKS02] P-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Eighth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2002. http ://www.cse.unsw.edu.au/ ∼qzhang/proceedings/research.html.

[TO94] J.D. Terwilliger and J. Ott. *Handbook of human genetic linkage*. Johns Hopkins University Press, Baltimore, June 1994. ISBN : 0801848032.

[Toi96] H. Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and Nandlal L. Sarda, editors, *In Proc. 1996 Int. Conf. Very Large Data Bases*, pages 134–145. Morgan Kaufman, 09 1996.

[TOO+00] H. Toivonen, P. Onkamo, V. Ollikainen, P. Sevon, H. Mannila, and J. Kere. Gene mapping by haplotype pattern matching. In *Proceedings, IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, pages 99–108, 2000.

[TOV+00] H. Toivonen, P. Onkamo, K. Vasko, V. Ollikainen, P. Sevon, H. Mannila, M. Herr, and J. Kere. Data mining applied to linkage disequilibrium mapping. *American Journal of Human Genetics*, 67 :133–145, 2000.

[UB03] S. Ujjin and P.J. Bentley. Particle swarm optimization recommender system. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003*, pages 124–131, Indianapolis, Indiana, USA, 2003.

[Unw00] A. Unwin. Visualisation for data mining. In *International Conference on Data Mining, Visualization and Statistical System*, 2000.

[VdJ93] H. Vafaie and K. de Jong. Robust feature selection algorithms. In IEEE Computer Society Press, editor, *In Proceedings of the Fifth Conference on Tools for Artificial Intelligence*, pages 356–363, Boston, 1993.

[Ven93] G. Venturini. SIA : a supervised inductive algorithm with genetic search for learning attributes based concepts. In P. Brazdil, editor, *European Conference on Machine Learning (ECML93)*, pages 280–296. Springer Verlag, Lecture Notes in Artificial Intelligence 667, 1993.

[Ven94] G. Venturini. Analyzing french justice with a genetic based inductive algorithm. *Applied Artificial Intelligence*, 8(4) :565–577, 1994. Special Issue on Real world applications of Machine Learning.

[Ven96] G. Venturini. Algorithmes génétiques et apprentissage. *Revue d'intelligence artificielle*, 10(2-3) :345–387, 1996.

[Ven97] G. Venturini. Apport des algorithmes génétiques à l'apprentissage et à l'optimisation. Technical report, E3I, Tours, France, 1997. Habilitation.

[Ver01] G. Vermeersch. Algorithmes génétique pour la bio-informatique. Master's thesis, USTL Lille 1, 2001.

[Wei99] G. M. Weiss. Timeweaver : a genetic algorithm for identifying predictive patterns in sequences of events. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 718–725, Orlando, Florida, USA, 1999. Morgan Kaufmann.

[WF01] J. C. Werner and T. C. Fogarty. Genetic programming applied to severe diseases diagnosis. In *Procedings Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2001)*, 2001.

[Whi89] Whitley. The GENITOR algorithm and selective pressure. In Morgan Kaufman, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pages 116–121, 1989.

[WL01] A. Wasson and T. Longemart. Etude des critères de qualité d'une règle d'association. Master's thesis, IUP GMI 3, Lille., 2001.

[WTL98] K. Wang, S. H. W. Tay, and B. Liu. Interestingness-based interval merger for numeric association rules. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 121–128. AAAI Press, 27–31 1998. New York, USA.

[WWT99] P.C. Wong, P. Whitney, and J. Thomas. Visualizing association rules for text mining. In *Proceedings IEEE Information Visualization 99*, October 1999.

[XDE+03] X. Xiao, E. Dow, R. Eberhart, Z. Ben Miled, and R. J. Oppelt. Gene clustering using self-organizing maps and particle swarm optimization. In *Second IEEE International Workshop on High Performance Computational Biology (HICOMB 2003)*, 2003. To appear.

[YH97] J. Yang and V. Honavar. Feature subset selection using A genetic algorithm. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997 : Proceedings of the Second Annual Conference*, pages 380–385, Stanford University, CA, USA, 1997. Morgan Kaufmann.

[YH98] J. Yang and V. Honoavar. *Feature Extraction, Construction and Selection : A data Mining Perspective*, chapter 1 : Feature Subset Selection Using a Genetic Algorithm, pages 117–136. H. Liu and H. Motoda Eds, massachussetts : kluwer academic publishers edition, 1998.

[Yu02] T-L. Yu. A survey of estimation of distribution algorithms. Technical report, IlliGAL, Illinois Genetic Algorithms Laboratory Department of General Engineering University of Illinois at Urbana-Champaign, USA, 2002.

[Zak99] M. J. Zaki. Parallel and distributed association mining : A survey. *IEEE Concurrency*, 7(4) :14–25, 1999.

[ZH00] M.J. Zaki and C.T. Ho, editors. *Large-Scale Parallel Data Mining*, volume 1759 of *State-of-the-Art Survey*. LNAI, 2000.

[ZK01] Y. Zhao and G. Karypis. Criterion functions for document clustering : Experiments and analysis. Technical Report Technical Report TR 01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, Minneapolis, MN, USA, 2001.

[ZOPL96] M. Zaki, M. Ogihara, S. Parthasarathy, and W. Li. Parallel data mining for association rules on shared-memory multiprocessors. Technical Report TR618, Department of Computer Science, University of Rochester, USA, 1996.

[ZPOL97] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. Heckerman, H. Mannila, D. Pregibon, R. Uthurusamy, and M. Park, editors, *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–296. AAAI Press, 12–15 1997.

[ZZ02] C. Zhang and S. Zhang. Association rules. *LNAI*, Association Rule Mining : Models and Algorithms(LNAI 2307) :25–46, 2002.

Annexe A

Glossaire des termes génétiques employés

Allèle : à un emplacement donné sur un chromosome, un même locus (emplacement) peut exister sous différentes formes appelées allèles. Les différences entre les allèles d'un même locus portent sur des variations de séquence.

Carte génétique : ordre des loci présentant des allèles (marqueurs gÃ©nÃ©tiques) et distance inter-marqueurs.

Chromosomes : éléments apparaissant, dans la cellule en cours de division, sous forme de bâtonnets qui subissent des processus complexes de dédoublement et de séparation. Les chromosomes sont les supports matériels des gènes et sont constitués d'une suite de nucléotide sur lesquels on trouve les marqueurs. Toutes les cellules d'une même espèce comportent $2n$ chromosomes ($n = 23$ chez l'homme), à l'exception des cellules sexuelles matures (les gamètes, spermatozoïdes et ovules), qui ne renferment que n chromosomes. Quand un spermatozoïde à n chromosomes (paternels) féconde un ovule à n chromosomes (maternels), il en résulte un oeuf fécondé, ou zygote, à $2n$ chromosomes appariés deux à deux (chromosomes "homologues"). Toutes les cellules issues de ce zygote, par un processus de division nommé mitose, comportent n chromosomes paternels et n chromosomes maternels. Seules les cellules précurseurs des cellules sexuelles subissent une division spéciale nommée méiose, qui aboutit à des gamètes à n chromosomes, après la séparation des chromosomes homologues.

Code génétique : chaque gène contient l'information nécessaire à la synthèse d'une protéine. L'information est contenue dans la séquence des quatre bases (A,T, C, G) qui s'alignent sur le fragment d'ADN. Une protéine est constituée de l'enchaînement linéaire d'acides aminés dont 20 types sont connus. La séquence de la protéine est déterminée par la séquence du gène qui la code. Il existe donc une correspondance entre les bases de l'ADN et les acides aminés. Celle-ci réside dans le code génétique : trois bases adjacentes sur l'ADN formant un triplet appelé codon spécifient un acide aminé. Les quatre bases prises trois à trois permettent de former 64 combinaisons soit un nombre supérieur à celui des 20 acides aminés. Il y a redondance du code génétique puisqu'à un

même acide aminé peut correspondre plusieurs codons (de 1 à 6). Ainsi, la méthionine, la valine et la sérine sont spécifiées respectivement par un codon (AUG), quatre codons (GUU, GUC,GUA et GUG) et six codons (UCU, UCC, UCA, UCG, AGU et AGC) (voir tableau) Le code génétique est universel, c'est à dire qu'il est le même de la bactérie à l'homme et s'applique également au règne végétal.

Epistasie : caractérise l'interaction de plusieurs gènes pour produire un phénotype donné.

Gène : unité d'information localisée sur un chromosome donné et constituée par un fragment d'ADN. Chaque gène dirige la production d'une ou plusieurs protéines et assure la transmission et l'expression d'un caractère donné. Le génome humain contient de l'ordre de 100.000 gènes répartis sur 23 paires de chromosomes.

Génome : ensemble du matériel génétique contenu dans une cellule. Il se présente sous la forme d'une longue molécule (1,50 m), l'ADN, lovée dans le noyau de chaque cellule de l'organisme. Etablir avec précision la situation et les fonctions des quelques 100.000 gènes du patrimoine génétique humain, tel est l'objectif du programme "génome humain".

Génotype / Phénotype : le génotype est constitué par l'ensemble des caractères héréditaires propres à un individu. La combinaison des deux gènes situés face à face sur les deux chromosomes homologues s'appelle le génotype. Le phénotype correspond à l'expression de ce patrimoine génétique dans un environnement donné. Il rend compte des caractéristiques anatomiques et physiologiques d'un individu. L'existence de gènes dominants et récessifs explique qu'à un même phénotype correspondent des génotypes différents.

Locus : endroit précis du génome. On fait en général l'hypothèse qu'il s'agit d'un segment assez petit pour qu'il n'y ait pas de recombinaisons internes.

Maladie multifactorielle : maladie liée à plusieurs facteurs. Les maladies multifactorielles reflètent 2 types de causalité qui interagissent entre elles : l'une génétique pouvant impliquer plusieurs gènes (multigéniques/polygénique), l'autre environnementale (la sÃ©dentaritÃ©, l'alimentation ou encore le tabagisme ...).

Maladie Polygénique : maladie génétique liée à plusieurs gènes.

Méiose (à ne pas confondre avec mitose) : processus de division des cellules précurseurs des gamètes, avec réduction par deux du nombre de chromosomes. La méiose aboutit à la formation des gamètes, ovules ou spermatozoïdes à n chromosomes.

Pénétrance d'un trait : un trait, un caractère, une maladie ont une pénétrance incomplète si leur probabilité de survenue est inférieure à 1, alors que le génotype responsable est présent. Cette probabilité est appelée taux de pénétrance.

Prévalence : dans une population donnée, nombre de sujets atteints de la maladie à un instant donné ou dans une période donnée. Elle augmente avec l'incidence mais aussi avec la durée de la maladie.

Recombinaison : échange physique de portions de chromosomes issus des gamètes parentaux lors de la méiose, entraînant un réarrangement des combinaisons génétiques dans les descendants.

Annexe C

Sélection d’attributs

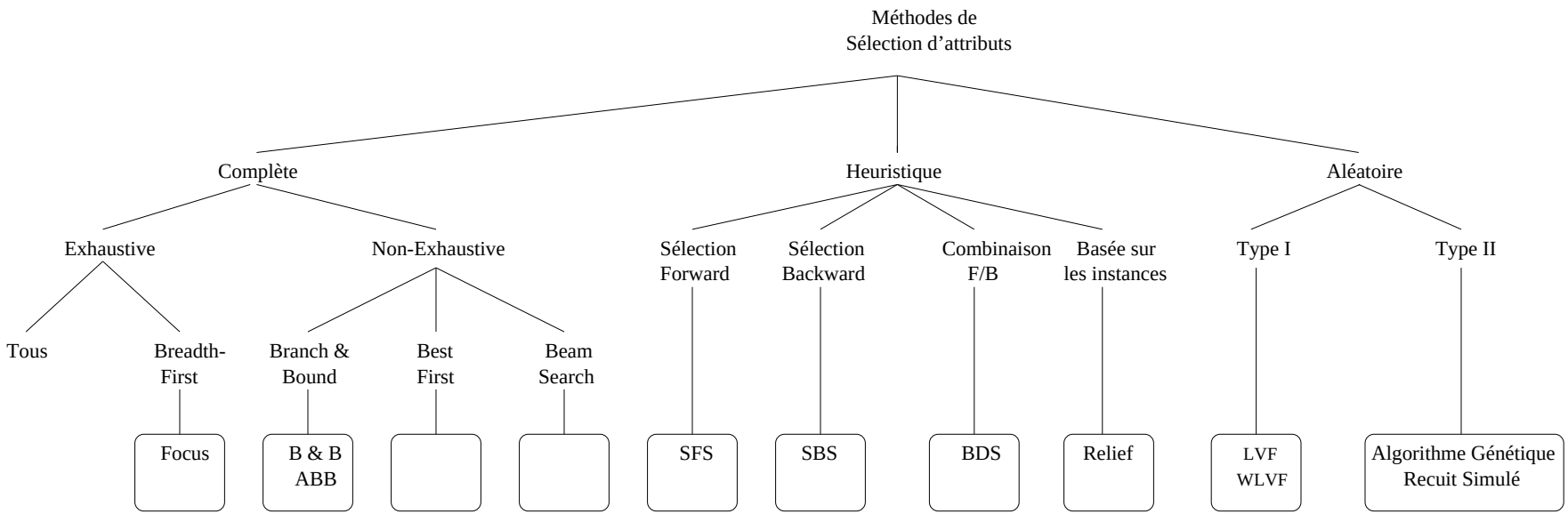


FIG. C.1 – Résumé de la hiérarchie des méthodes

Article	Objet	Méthode	Base	# Classe	# att.	# att. sélectionnés (%)		
						C4.5	NB	ID3
R. Kohavi et G. John[KJ98]	Classification	BFS (Wrapper)	breast Cancer	2	10	3.9 (39.0)	5.9 (59.0)	5.3 (53.0)
			cleve		13	5.3 (40.7)	7.9 (60.7)	4.6 (35.3)
			crx		15	7.7 (51.3)	9.1 (60.6)	7.7 (51.3)
			DNA	39	180	12 (6.66)	48 (26.66)	36 (20.0)
			horse-colic		22	4.2 (19.1)	6.1 (27.7)	7.2 (32.7)
			Pima		8	4.8 (60.0)	4.4 (55.0)	5.7 (71.2)
			sick-euthyroid		25	3 (12.0)	3 (12.0)	4 (16.0)
			soybean-large		35	17.1 (48.8)	16.7 (47.7)	17.7 (50.5)
			Corral		6	2 (33.3)	5 (83.3)	4 (66.6)
			m-of-n-3-7-10		10	6 (60.0)	7 (70.0)	7 (70.0)
			Monk1	2	6	3 (50.0)	4 (66.6)	3 (50.0)
			Monk2-local		17	6 (35.3)	5 (29.4)	6 (35.3)
			Monk2	2	6	0 (0.0)	0 (0.0)	3 (50.0)
			Monk	2	6	2 (33.3)	2 (33.3)	2 (33.3)
C. Guerra-Salcedo et D. Whitley [GSW99]	Création d'ensembles					En moyenne		
		CHC-EDT	DNA	39	180	11.24	±	2.13
			LandSat	6	36	12.6	±	1.89
			Segment	7	19	3.6	±	0.728
		CHC-KMA	DNA	39	180	16.26	±	3.5
			LandSat	6	36	11.04	±	2.16
			Segment	7	19	5.9	±	0.886

Article	Objet	Méthode	Base	# Classe	# att.	# att. sélectionnés (%)	
						Moyenne	Optimal
K. Chen et H. Liu [CL99b]	Classification	WLVF	Soybean	15	35	9.4	9
			Par5+5	2	10	5.0	5
			Mush	2	22	4.2	4
			Vote	2	16	4.0	4
			Monk 1	2	6	3.0	3
			Monk 2	2	6	6.0	6
			Monk 3	2	6	4.0	4
			Hepat	2	19	3.0	3
			Led24	10	24	5.0	5
			Derma'y	6	34	6.3	6
			Par10+10	10	20	10.0	10
			Zoo	7	16	4.0	4
			TicTacToe	2	9	7.0	7
Guerra-Salcedo et D. Withley [GSW98]	Classification					Meilleur	En Moyenne
		Genesis+EDT	LandSat	6	36	21	20.7
			Cloud	10	204	103	105.2
		CHC + EDT	LanSat	6	36	16	17.4
			Cloud	10	204	99	96.4

TAB. C.1 – Comparatifs des résultats obtenus par différentes méthodes d'extraction d'attributs pour la classification.

Article	Objet	Méthode	Base	K	# Att.	# Att. Sél.	Accuracy	# Sous-Ens. Testé	Temps
Devaney et al. [DR97]	Clustering	COBWEB (relevant)	Heart Disease		75	13	0.755	1	203.33
		COBWEB (all)			75	75	0.567	1	923.33
		COBWEB-FS			75	18	0.677	577.33	31560.67
		AICC-FS			75	16.33	0.733	659.33	5902.67
		COBWEB-BS			75	20.67	0.813	1064.00	280890.00
		AICC-BS			75	19.33	0.784	1128.33	53521.00
		COBWEB (relevant)	LED		24	7	1	1	66.33
		COBWEB (all)			24	24	0.77	1	224.00
		COBWEB-FS			24	6.67	0.93	158.33	11451.00
		AICC-FS			24	7	1	164.00	5322.00
		COBWEB-BS			24	12	0.88	204.67	50482.00
		AICC-FS			24	8	0.89	280.00	14812.00

TAB. C.2 – Comparatifs des résultats obtenus par COBWEB et AICC.

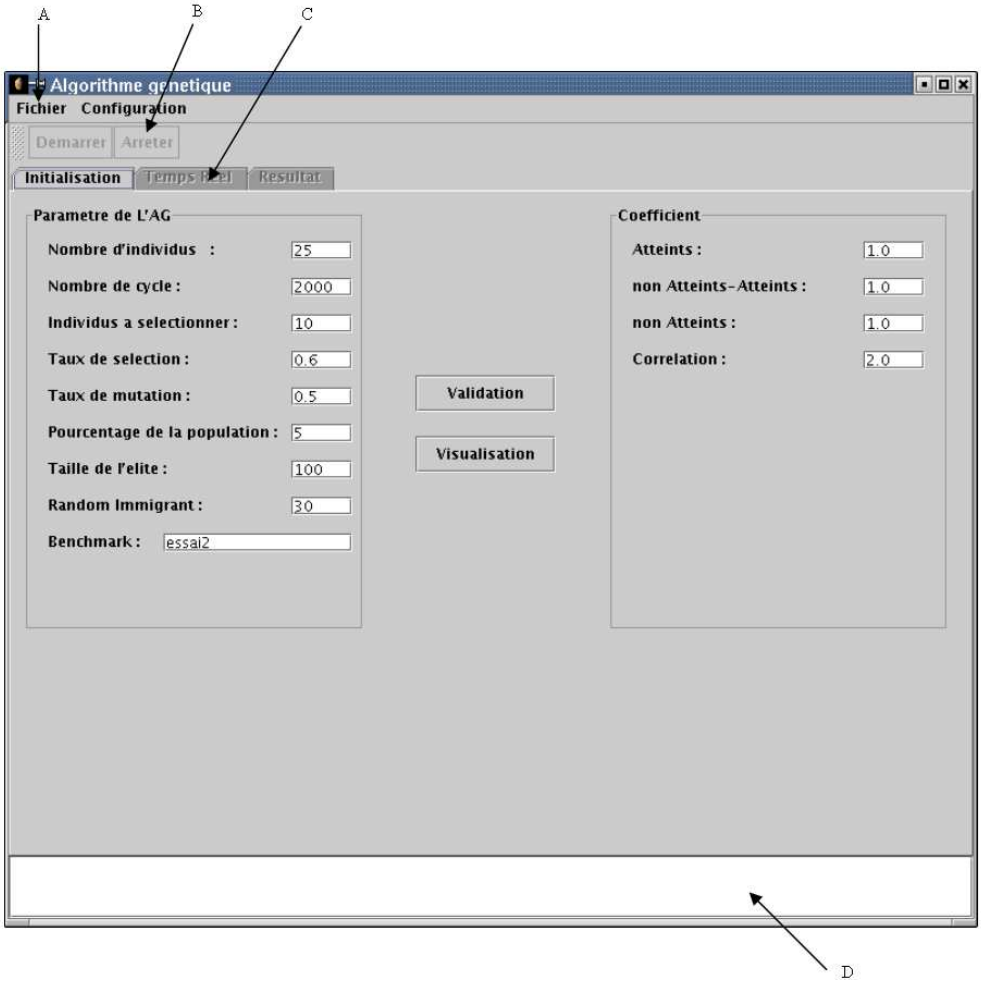


FIG. C.2 – Onglet de paramétrage de l'interface graphique. La fenêtre graphique contient différents boutons et panneaux : le bouton B et l'onglet C qui s'activent lorsque le jeu de données est validé, le panneau contient lui les messages produits par l'algorithme génétique.

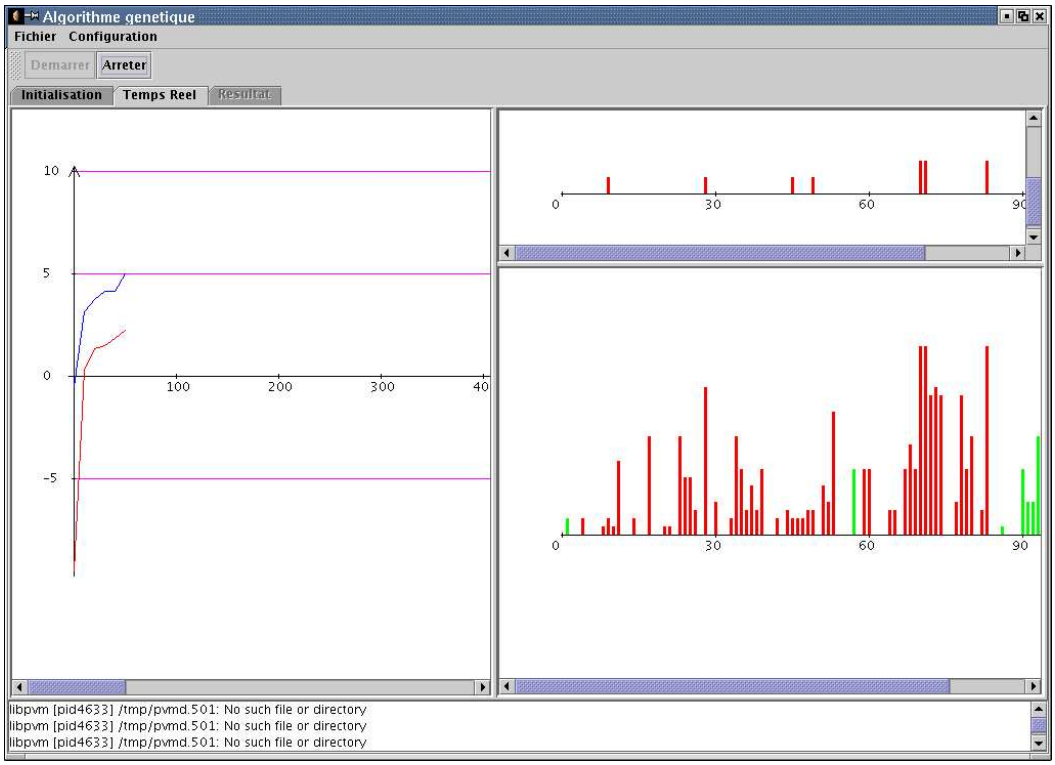


FIG. C.3 – Premières itérations de l'algorithme génétique.

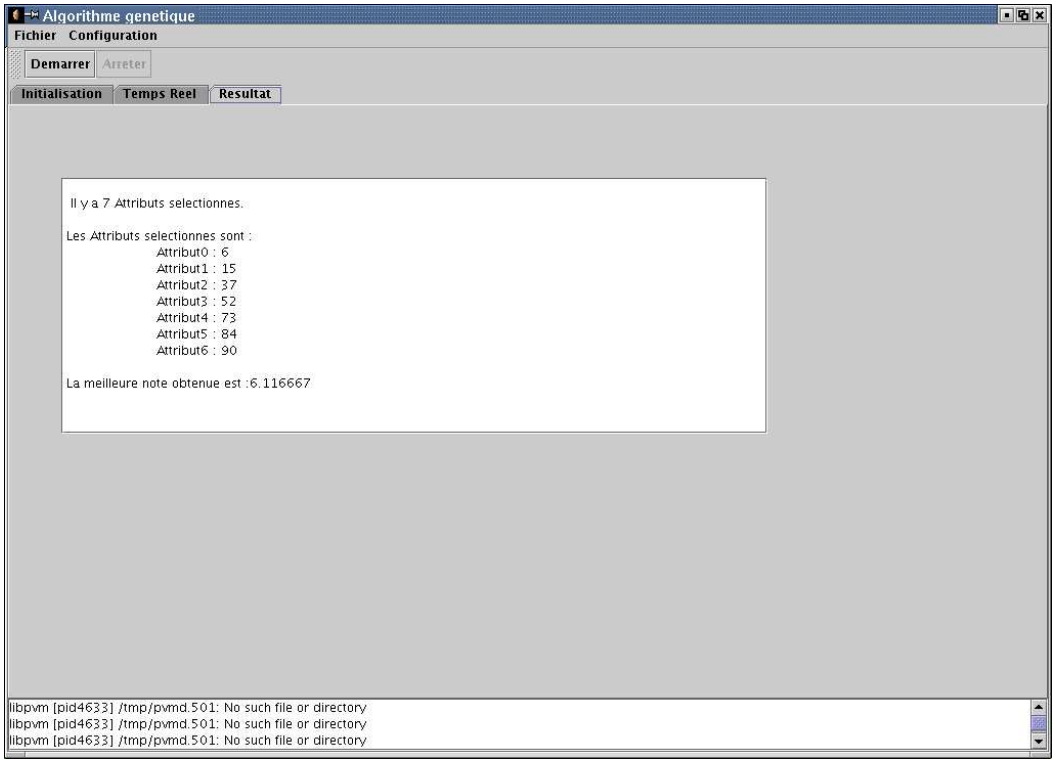


FIG. C.4 – Le dernier onglet présente les résultats obtenus sur les fichiers de benchmark.

Annexe D

Descriptif des jeux de données pour le clustering

Cette annexe présente les principales bases de tests sur lesquelles ont été testées les algorithmes présentés dans le chapitre 5.6. Elles ont été choisies pour démontrer l'adaptabilité de l'algorithme aux types de données et à la taille des problèmes.

D.1 AD

Les jeux de données artificiels (Artificial datasets AD) ont été conçus par Bandyopadhyay [BM02]. Une représentation graphique est donnée figure D.1.

D.1.1 AD_5_2

AD_5_2 est un jeu de données bi-dimensionnel composé de 250 instances et sur lequel on peut remarquer que les cinq clusters se recouvrent fortement (figure D.1-gauche).

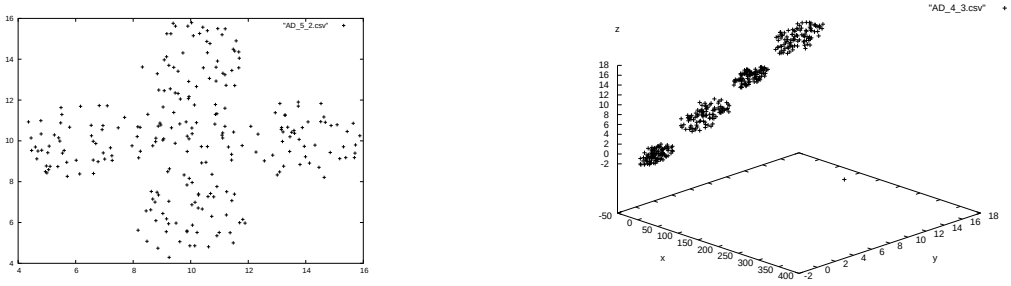


FIG. D.1 – Les jeux de données AD_5_2 et AD_4_3.

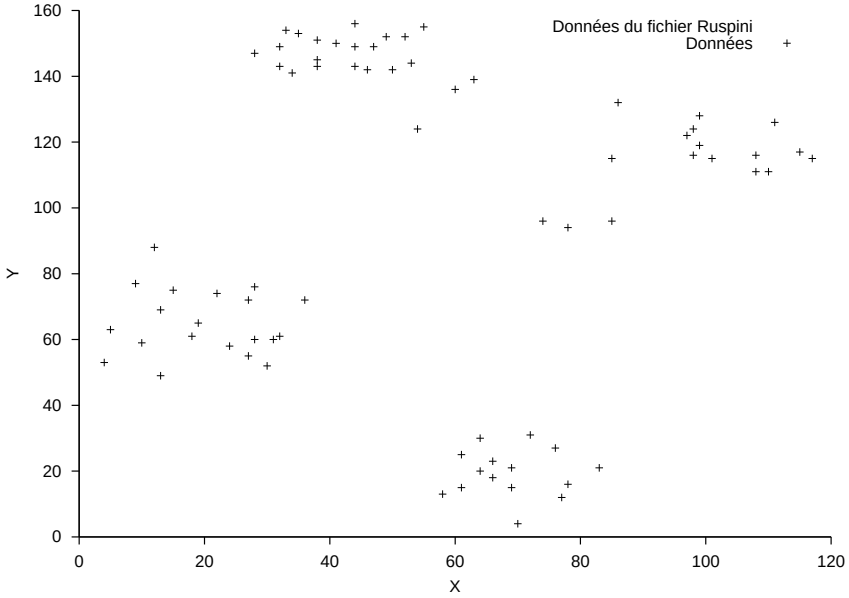


FIG. D.2 – Les données du fichier ruspini.

D.1.2 AD_4_3

AD_4_3 est un jeu de données tri-dimensionnel composé de 402 instances. Ce jeu de données contient quelques instances bruitées situées exactement entre deux clusters distincts (figure D.1-droite).

D.2 Ruspini

E. H. Ruspini (1970) : Numerical methods for fuzzy clustering. Inform. Sci., 2, 319–350.
Les données Ruspini consistent en 75 points répartis en quatre groupes. Ce jeu de données est populaire pour illustrer les techniques de clustering.
Pour les 75 points, les données disponibles sont les coordonnées x et y. La figure D.2 illustre la répartition des points sur le plan.

	Mean	Standard Deviation
x	54.88	30.5025
y	92.0267	48.7026

TAB. D.1 – Statistiques descriptives de la base Ruspini.

D.3 Iris

Iris Plant Database disponible sur le site de l'UCI [BM98].
A l'origine de cette base de données, les travaux de R.A. Fisher "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936).
Ces données sont souvent utilisées en classification. Il y a 3 classes d'Iris à découvrir : Iris Setosa, Iris Versicolour et Iris Virginica. La base de données contient 150 instances réparties à égalité dans chaque classe (50 par classe).
Il y a quatre attributs numériques : sepal length (longueur du sépal) en cm, sepal width (largeur du sépal) en cm, petal length (longueur du pétal) en cm et petal width (largeur du pétal) en cm.
Le tableau D.2 donne des indications sur les données de cette base.

	Min	Max	Mean	SD	Class Correlation
sepal length	4.3	7.9	5.84	0.83	0.7826
sepal width	2.0	4.4	3.05	0.43	-0.4194
petal length	1.0	6.9	3.76	1.76	0.9490
petal width	0.1	2.5	1.20	0.76	0.9565

TAB. D.2 – Statistiques descriptives de la base Iris

D.4 Breast Cancer Winconsin (Cancer)

Le jeu de données Wisconsin Breast Cancer Database (Cancer) contient les informations médicales de 683 cas cliniques relatifs au cancer du sein classés comme bénin ou malin. Les cas cliniques sont décrits par 9 attributs numériques [MW90].

1. Clump Thickness
2. Uniformity of Cell Size
3. Uniformity of Cell Shape
4. Shape Marginal Adhesion
5. Single Epithelial Cell Size
6. Bare Nuclei
7. Bland Chromatin
8. Normal Nucleoli
9. Mitoses

D.5 Diabetes

La base de données Pima Indians Diabetes Database provient du site de l'UCI et a été constituée par l'Institut national des maladies du diabète, de la digestion et du foie. L'objectif est de

réaliser un diagnostic du diabète sur une population vivant près de Phoenix, Arizona aux USA.
Il y a 768 patients (donc instances) qui sont toutes des femmes de plus de 21 ans et provenant toutes des tribus indiennes Pima. Les huit attributs numériques observés sont :

1. Nombre de grossesses
2. Concentration du plasma en glucose après un test de tolérance au glucose de 2 heures
3. Pression diastolique du sang (mm Hg)
4. Epaisseur de la peau au niveau du triceps (mm)
5. Taux d'insuline au bout de 2 heures (mu U/ml)
6. Body mass index (poids en kg/(taille in m)^2)
7. Fonction pédigrée du diabète
8. Age (années)

Le tableau D.3 donne des indications sur les données de cette base.

	Mean	**Standard Deviation**
1.	3.8	3.4
2.	120.9	32.0
3.	69.1	19.4
4.	20.5	16.0
5.	79.8	115.2
6.	32.0	7.9
7.	0.5	0.3
8.	33.2	11.8

TAB. D.3 – Statistiques descriptives de la base Diabete.

D.6 Vote

Ce jeu de données contient les enregistements des votes du congrès américain de 1984. Source : Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL : Congressional Quarterly Inc.
Ce jeu de donnés contient les votes de chaque représentant au congrès pour 16 points clés donnés par le CQA. A l'origine le CQA liste neuf types de réponses : voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).
Il y a 435 instances (267 démocrates, 168 républicains) pour 16 attributs booléens. Attributs :

1. handicapped-infants : 2 (y,n)
2. water-project-cost-sharing : 2 (y,n)
3. adoption-of-the-budget-resolution : 2 (y,n)
4. physician-fee-freeze : 2 (y,n)

Attribut	Valeur manquante
1	0
2	12
3	48
4	11
5	11
6	15
7	11
8	14
9	15
10	22
11	7
12	21
13	31
14	25
15	17
16	28

5. el-salvador-aid : 2 (y,n)
6. religious-groups-in-schools : 2 (y,n)
7. anti-satellite-test-ban : 2 (y,n)
8. aid-to-nicaraguan-contras : 2 (y,n)
9. mx-missile : 2 (y,n)
10. immigration : 2 (y,n)
11. synfuels-corporation-cutback : 2 (y,n)
12. education-spending : 2 (y,n)
13. superfund-right-to-sue : 2 (y,n)
14. crime : 2 (y,n)
15. duty-free-exports : 2 (y,n)
16. export-administration-act-south-africa : 2 (y,n)

Le jeu de données contient des valeurs manquantes notées “ ?” mais pouvant être informative.

D.7 Breast Cancer

Cette base de données relative au cancer du sein a été mise à disposition par l’université du Wisconsin par Dr. William H. Wolberg.
O.L. Mangasarian and W. H. Wolberg : "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 – 18.

Le jeu de données est composé de 699 instances pour 10 attributs.

Num.	Attribut	Domaine
1	Sample code number	id number
2	Clump Thickness	1 - 10
3	Uniformity of Cell Size	1 - 10
4	Uniformity of Cell Shape	1 - 10
5	Marginal Adhesion	1 - 10
6	Single Epithelial Cell Size	1 - 10
7	Bare Nuclei	1 - 10
8	Bland Chromatin	1 - 10
9	Normal Nucleoli	1 - 10
10	Mitoses	1 - 10

TAB. D.4 – Présentation des attributs et de leur domaine pour le jeu de données Breast cancer.

Il y a 16 valeurs manquantes.

D.8 Lung cancer

Le jeu de données “Lung cancer” (cancer du poumon) est disponible sur le web et a été publié dans Hong, Z.Q. and Yang, J.Y. "Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane", Pattern Recognition, Vol. 24, No. 4, pp. 317-324, 1991.
Ce jeu de données décrit trois types de pathologie du cancer du foie. Il contient des données manquantes pour deux des attributs. Le jeu contient 32 instances pour 56 attributs descriptifs. Tous les attributs sont nominaux et prennent leur valeur parmi les entiers de 0 à 3.