Etude des Algorithmes génétiques et application aux données de protéomique

Christelle Reynès

A C A D É M I E D E M O N T P E L L I E R

UNIVERSITÉ MONTPELLIER I

— UFR DES SCIENCES PHARMACEUTIQUES —

Etude des Algorithmes Génétiques et Application aux Données de Protéomique

Thèse présentée pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITE MONTPELLIER I

ECOLE DOCTORALE : *Information, Structure et Systèmes*
FORMATION DOCTORALE : *Biostatistique*
N° de DISCIPLINE : *42* N° CNU : *26*
SPECIALITE : *Statistique*

par

Christelle REYNES

soutenue publiquement le
20 juin 2007

devant le jury

M. Gilles DUCHARME	Professeur, Université Montpellier II	
M. Jean-Pierre DURBEC	Professeur, Université de Luminy	
M. Yann GUEDON	Chercheur, CIRAD	
M. Nicolas MOLINARI	Maître de Conférences, Université Montpellier I	Co-directeur de thèse
M. Bernard PRUM	Professeur, Université d'Evry	Rapporteur
M. Robert SABATIER	Professeur, Université Montpellier I	Directeur de thèse
M. Bernard YCART	Professeur, Université Grenoble I	Rapporteur

Résumé
Les algorithmes génétiques sont des méthodes d'optimisation destinées à des problèmes complexes. Ils peuvent jouer un rôle intéressant dans le cadre de la protéomique. Cette discipline est assez récente, elle étudie le patrimoine en protéines des individus. Elle produit des données de grande dimension.
La première partie aborde l'histoire, le fonctionnement des algorithmes génétiques et certains résultats théoriques. La partie suivante détaille la mise au point d'un tel algorithme pour la sélection de biomarqueurs en spectrométrie de masse et l'alignement de gels d'électrophorèse 2D. Cette partie met en évidence la difficulté de construction du critère à optimiser. La dernière partie aborde des résultats théoriques. La convergence des algorithmes génétiques avec élitisme est démontrée dans le cas non homogène et de mutations dirigées. Nous avons ensuite construit un critère de convergence alliant fondements théoriques et applicabilité, basé sur les occurrences de la solution localement optimale. Enfin, l'efficacité de l'introduction d'événements catastrophiques dans la résolution pratique de certains problèmes de convergence est montrée.

Mots clés : Algorithme génétique, protéomique, convergence, SELDI, électrophorèse 2D.

Abstract
Genetic Algorithms are optimization methods aiming at solving complex problems. They are likely to play an interesting role in proteomics. This discipline is a quite new one which studies the proteins in individuals. It provides high dimension data.
The first part deals with history, working of genetic algorithms and introduces some theoretical results. In the next part the building of a genetic algorithm is presented to solve biomarker selection in mass spectrometry and 2D electrophoresis gels alignment. This part focuses on the difficulty to choose an appropriate criterion to optimize. The last part deals with theoretical results. Convergence of elitist genetic algorithms is proved for non homogeneous case and orientated mutations. Then we built a convergence criterion mixing theoretical basements and appliability, which is based on occurrences of the locally optimal solution. Finally, the efficiency of introducing catastrophic events to avoid some convergence problems is shown.

Keywords : Genetic algorithms, proteomics, convergence, SELDI, 2D electrophoresis.

Remerciements

La première personne que je voudrais remercier est celui sans lequel ce travail n'aurait jamais existé, mon directeur de thèse, Robert Sabatier. Même si plusieurs personnes ont œuvré pour me pousser dans cette direction, c'est lui qui a su trouver les arguments qui ont fait pencher la balance, définitivement. Par la suite, il a fallu qu'il assume et qu'il me supporte, tâche difficile dont il s'est acquitté avec brio. Il a été un directeur de thèse présent, attentif, extrêmement impliqué et fourmillant d'idées. Malheureusement, il n'écoute que de la musique classique !... Pour tout, merci.

Lors de mon DEA une autre personne, Nicolas Molinari, s'est beaucoup démenée pour me pousser à me lancer dans cette entreprise qu'est la thèse. Cela l'a naturellement conduit à être mon co-directeur de thèse. Je le remercie pour son soutien même si notre éloignement géographique a empêché une étroite collaboration. Il m'a accompagnée dans tous les congrès, a subi de nombreuses relectures de mes écrits *fleuris* et a su m'apporter de nombreux contacts dans le monde des biologistes.

Je tiens à exprimer ma gratitude au Professeur Bernard Prum qui a accepté très rapidement d'être rapporteur de ma thèse et qui, dans un temps record, m'a fourni une correction TRES détaillée de mon manuscrit. Ses remarques concernant le fond ont été très précieuses et sa maîtrise de la langue française a forcé mon admiration (ce qui n'est pas si courant chez les matheux, n'est-ce pas Thomas ?).

Je remercie profondément le Professeur Bernard Ycart qui a également tenu le rôle difficile de rapporteur. Ses remarques ont été d'une grande précision et surtout d'un très grand intérêt pour l'amélioration de mon travail mais aussi pour la perspective de travaux futurs.

Pour ce qui est de la composition de mon jury, j'ai également tenu à la présence de Yann Guédon dont les cours en DEA m'avaient réellement intéressée et m'ont reconduit naturellement vers lui quand j'ai commencé à me frotter aux chaînes de Markov. Malgré mes nombreuses sollicitations, son accueil a toujours été chaleureux et attentif et ses réponses et orientations tout à fait judicieuses.

Le professeur Gilles Ducharme a donné le premier *coup de pouce* à ma thèse. Il a en effet pris la peine de me téléphoner pour me convaincre de modifier mon choix de stage de DEA. Je sais maintenant combien il avait raison... C'est pourquoi je suis particulièrement heureuse de le compter parmi les membres de mon jury.

Enfin, le Professeur Durbec a spontanément accepté de faire partie de ce jury, montrant une fois de plus sa grande ouverture d'esprit pour des problèmes statistiques lui permettant de renouer avec ses précédents centres d'intérêt. Je l'en remercie énormément.

Evidemment, et comme ce sera détaillé dans les pages à venir (pour ceux qui ne s'arrêteront pas aux remerciements !), ce travail est le fruit d'une étroite collaboration avec les équipes de biologistes et surtout celle du Professeur Sylvain Lehmann qui, aidé de Stéphane Roche

et Laurent Tiers, m'a initiée aux techniques de la protéomique.

Je n'oublie pas Ludovic Menneteau vers qui je me suis tournée dans un moment de grand manque d'inspiration et qui m'a aiguillée vers Pierre Pudlo que je remercie beaucoup pour sa disponibilité et les très précieux conseils qu'il m'a donnés dans le domaine des chaînes de Markov.

Ma gratitude va également à l'ensemble de l'équipe du Professeur Luc Maury, qui m'a accueillie pendant trois ans et demi et fourni les conditions de travail qui m'ont permis d'avancer. Je pense particulièrement à Laurent Vachoud car d'une part nous avons partagé de bonnes parties de rigolade (notamment grâce aux TDs) et d'autre part il a beaucoup contribué à l'ambiance chaleureuse du labo (notamment grâce à B.F.). Un grand merci également à Michèle Delalonde qui dès son arrivée s'est intéressée à mon travail et m'a soutenue dans les moments difficiles.

Je n'oublie évidemment pas mes collègues thésards avec lesquels j'ai beaucoup échangé et partagé pendant ces quatre dernières années : Thomas Verron avec lequel j'ai passé d'excellents moments et qui sait par quoi je suis passée puisque nous avons subi le même directeur de thèse (!...), Christel Castelli qui m'a supportée pendant les congrès et bien plus, qui a compati à mes soucis et partagé de grands moments, Yohann Foucher qui m'a sortie de bien des guêpiers informatiques et qui s'est souvent sacrifié pour faire équipe avec moi aux fléchettes, Vanessa Rousseau qui est arrivée à point nommé pour me prendre de temps en temps ma dernière place à ce jeu et enfin Myrtille Vivien, même si elle n'est plus thésarde depuis longtemps, qui m'avait chauffé la place au labo (et nous savons toutes les deux que c'est nécessaire...) et avec laquelle j'ai partagé de très bons moments pour l'organisation du congrès Agrostat 2006.

Evidemment, je pense à ma famille qui m'a soutenue dans tous mes choix, même les plus difficiles, et qui a su me soutenir dans les pires moments.

Beaucoup d'esprits distingués paraissent ne pas pouvoir accepter ni même comprendre que d'une source de bruit la sélection ait pu, à elle seule, tirer toutes les musiques de la biosphère.

Jacques Monod (1970)

Table des matières

Notations

$\mathbf{X}$, $\mathbf{Y}$, $\mathbf{\Pi}$, ... : les lettres majuscules en caractères gras désignent des matrices.
x_{ij} : si $\mathbf{X}$ est une matrice, x_{ij} est l'élément de $\mathbf{X}$ se trouvant à l'intersection de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne.
$\mathbf{a}$, $\mathbf{c}$, ... : les lettres minuscules en caractères gras désignent des vecteurs.
x_i : si $\mathbf{x}$ est un vecteur, x_i désigne son $i^{\text{ème}}$ élément.
n, π, N, ... : les lettres minuscules et majuscules en caractères non gras désignent des réels, sauf indication contraire.
$\mathbf{I}_k$: matrice identité de dimension $k \times k$.
$\mathbf{0}_k$: vecteur colonne de longueur k ne comportant que des zéros.
$\mathbf{1}_k$: vecteur colonne de longueur k ne comportant que des uns.
$\mathbf{X}'$: matrice transposée de la matrice $\mathbf{X}$.
$tr[\mathbf{X}]$: trace de la matrice $\mathbf{X}$.
$\mathcal{A}$, $\mathcal{X}$, ... : les majuscules calligraphiques désignent des ensembles sauf $\mathcal{N}$ qui désigne un nombre de solutions et $\mathcal{V}$ qui désigne un voisinage.
$x^1, x^2, \ldots$: éléments de l'ensemble $\mathcal{X}$.
$x^{(1)}, x^{(2)}, \ldots, x^{(n)}$: la permutation de $x^1, x^2, \ldots, x^n$ telle que ses éléments sont ordonnés au sens d'un critère à préciser.
$\mathcal{X} \setminus \mathcal{A}$: complémentaire de la partie $\mathcal{A}$ de l'ensemble $\mathcal{X}$.
$\{x : E(x)\}$: ensemble des éléments x satisfaisant à la condition $E(x)$.
$\mathbf{Pr}[E]$: probabilité de l'événement E.
$\mathbf{Pr}[E|F]$: probabilité conditionnelle de l'événement E relative à l'événement F (probabilité pour que E soit réalisé sachant que F l'est).
$\mathbb{E}[X]$: espérance de la variable aléatoire X.
$\binom{n}{k}$: cardinal de l'ensemble des combinaisons de k éléments d'un ensemble à n éléments ($\binom{n}{k} = \frac{n!}{k!(n-k)!}$).
$\mathbb{I}\{E\}$: fonction indicatrice qui vaut 1 si l'événement E est réalisé et 0 sinon (sauf p.50 où la notation est légèrement différente pour s'adapter au contexte).
f : fonction dite de *fitness* d'un ensemble $\mathcal{X}$ dans $\mathbb{R}$.
$\frac{\partial f(x,y)}{\partial x}$ dérivée partielle de la fonction f par rapport à x.
$\otimes$ désigne la multiplication de vecteurs élément par élément.
$\oplus$ désigne l'addition de vecteurs élément par élément.

Acronymes

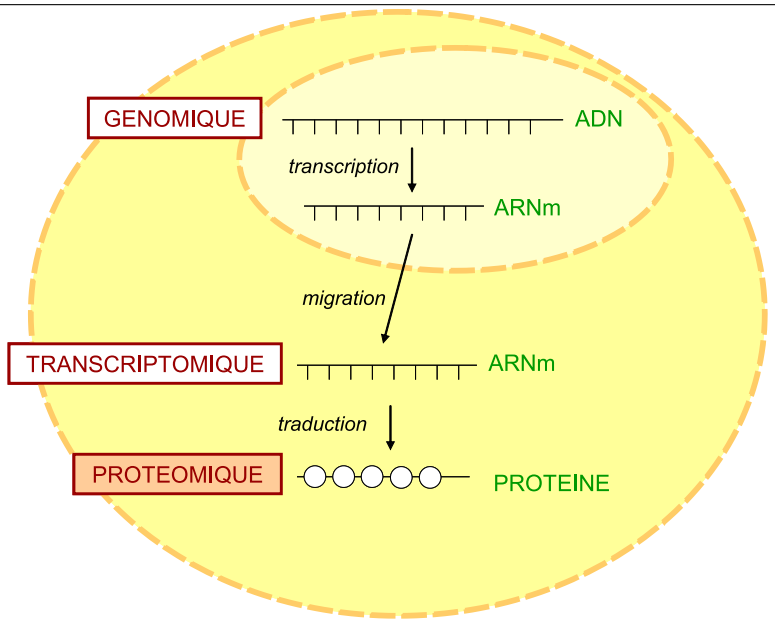
ACO : Ant Colony Optimization
ACP : Analyse en Composantes Principales
ADN : Acide DésoxyriboNucléique
AG : Algorithme Génétique
AG-AFD : Algorithme Génétique - Analyse Factorielle Discriminante
ARNm : Acide RiboNucléique Messager
CJD : Creutzfeld-Jacob Disease
GA-SCA : Genetic Algorithm - Simple Component Analysis
GLS : Guided Local Search
GRASP : Greedy Randomized Adaptive Search Procedures
HR : Hypothèse de Récurrence
k-FCV : k-Fold Cross Validation
k-ppv : k plus proches voisins
NA : Not Available
NIRS : Near InfraRed Spectrometry (SPIR en français)
PLS : Partial Least Squares
RCL : Restrictive Candidate List
SA : Simulated Annealing
SCA : Simple Component Analysis
SELDI-TOF : Surface Enhanced Laser Desorption Ionisation - Time Of Flight (noté indifféremment SELDI).
SPIR : Spectrométrie Proche Infra Rouge (NIRS en anglais)
SS : Scatter Search
SVM : Support Vecteur Machine
TS : Tabu Search
VIP : Variable Importance in Prediction
VNS : Variable Neighbourhood Search
2D-PAGE : 2-Dimensional - PolyAcrylamide Gel Electrophoresis

Introduction : les Algorithmes génétiques, un outil efficace entre biologie et mathématiques

La protéomique, une discipline idéale pour une collaboration biologie/statistique

Une fois n'est pas coutume, nous allons commencer cette étude statistique par une petite introduction biologique qui permettra de comprendre l'intérêt de ce travail. Pour rassurer tout le monde, nous allons commencer par une petite illustration en couleur (Fig. 1) qui nous permettra de mettre en place les différents concepts simples nécessaires à cette compréhension.
Nous l'aurons bien compris grâce à cette figure, la biologie moderne est la biologie des -*omiques*. Ce sont toutes trois (gén*omique*, transcript*omique* et protéo*mique*) des disciplines récentes et spécialement pour celle qui nous intéresse, la protéomique, dont le nom est apparu pour la première fois dans un article scientifique en 1997 (James, 1997). D'après le graphique, on pourrait croire que, les molécules provenant les unes des autres, l'étude de l'une est équivalente à l'étude de l'autre. Ce n'est pas le cas. En effet, dans le cadre de la génomique, on étudie le patrimoine en gènes (en ADN) des individus. Cette discipline a donné lieu à de nombreuses études statistiques (Muri-Majoube & Prum, 2002). Cependant, si le patrimoine génétique est variable d'un individu à l'autre, il est le même dans toutes les cellules d'un individu, quelles que soient les conditions. Une telle information est intéressante, notamment pour trouver un éventuel gène qui prédispose à telle ou telle pathologie.
Toutefois, il est plus intéressant de travailler sur les produits de la transcription de l'ADN, les ARNm (pour messager). En effet, tous les gènes d'une cellule ne sont pas transcrits dans toutes les cellules, chaque cellule, en fonction de sa spécialisation (cellule du muscle cardiaque, cellule de l'œil,...), *exprime* un certain nombre de gènes qui peut varier en fonction des conditions. L'information produite par les ARNm est donc beaucoup plus spécifique, elle permet également de constater des évolutions en cas de changement de conditions (stress hydrique pour les cellules d'*Arabidopsis Thaliana*, développement d'une tumeur dans une glande mammaire humaine, maturation de la viande de porc après l'abattage,...). L'étude est donc beaucoup plus dynamique et les applications sont plus variées. Mais on peut faire encore mieux : la protéomique ! Les protéines résultent de la traduction de l'ARNm. Ce sont elles les molécules fonctionnelles, celles qui vont intervenir dans les réactions chimiques, celles qui vont propager les informations,... bref celles qui font fonctionner les organismes. On passera ici sur le débat de *qui de la poule ou de l'œuf* dans le cas des protéines et des

Fig. 1 De l'ADN aux protéines : trois molécules, trois étapes, trois disciplines.



acides nucléiques (ADN/ARN), sachant que dans les organismes actuels (et ce depuis des millions d'années), les protéines sont indispensables à la synthèse de l'ADN et que l'ADN possède l'information qui permet de synthétiser les protéines...

En plus d'être fonctionnelles, les protéines ont l'avantage sur les ARN d'être plus variées. En effet, même si les ARN subissent des modifications après transcription, elles ne sont rien par rapport aux transformations qui touchent les protéines après leur synthèse. Ces modifications, très diverses, peuvent être minimes d'un point de vue chimique mais changer radicalement la fonction de la protéine affectée. L'étude de ces protéines est donc, d'un point de vue chimique, la photographie la mieux résolue de l'état d'un tissu, dans un organisme, pour des conditions données. Evidemment, ce seront surtout des changements de conditions qui nous intéresseront, avec de possibles apparitions/disparitions de protéines ou modifications de leur quantité dans les cellules.
Après ce petit résumé simpliste, une question viendra sans doute au lecteur curieux : mais pourquoi donc, alors que la génomique semble être la moins intéressante des trois, entend-on toujours parler de cette discipline et peu de la protéomique qui semble supérieure ? On peut donner deux raisons principales à cela :

- l'étude du patrimoine génétique des individus est certes limitée en termes d'application mais son analyse est assez simple, étant donné que l'on peut l'étudier n'importe quand et n'importe où dans l'organisme, on aura toujours les mêmes données, alors que dans le cas de la protéomique, le choix du tissu à étudier et du temps de prélèvement seront critiques pour les résultats,
- les techniques d'étude des acides nucléiques sont bien plus simples, plus calibrées, plus reproductibles que celles qui permettent d'étudier les protéines.

Ce dernier point est critique et nous concerne particulièrement. En effet, étant donnée la spécificité des résultats concernant les protéines, on comprend rapidement que, pour comparer deux échantillons, pour deux conditions intéressantes, il va falloir, dans la mesure du

possible, se débarrasser de toute la variabilité qui ne nous intéresse pas, même pour un seul tissu d'un seul individu. On aura donc tendance à vouloir réaliser des répétitions. De plus, nous avons évoqué le fait qu'une modification mineure d'une protéine peut avoir de très importantes répercussions, les biologistes vont donc naturellement chercher à avoir les analyses les plus complètes et précises possibles, ce qui conduit à de très gros jeux de données. Rajoutons également que les techniques de quantification des protéines sont beaucoup moins précises que celles qui concernent les acides nucléiques et on comprend rapidement l'intérêt de l'intrusion des statisticiens dans le monde des biologistes qui s'attaquent à la protéomique.
Avant de passer aux développements plus statistiques, profitons de cette occasion pour souligner, dans ce type d'étude, l'importance primordiale d'une réelle communication entre biologistes et statisticiens. Nous l'avons vu, les données de protéomique sont très complexes, elles ne peuvent donc pas se contenter d'un traitement par les méthodes classiques généralement bien maîtrisées par les biologistes. Ces données nécessitent le plus souvent la mise en place de méthodes spécifiques qui, là, échappent aux compétences de la plupart des biologistes : *chacun son boulot !* Cependant, il ne faudrait pas prendre ici, le raccourci qui est à l'origine de la méfiance de nombre de biologistes envers les statisticiens et qui consisterait à penser que les biologistes ne font que produire des données et que les statisticiens détiennent, eux, les clefs de l'analyse de ces données et donc les résultats. C'est totalement faux et hautement préjudiciable à une bonne qualité de l'analyse des données.
En effet, lors de la mise au point d'une nouvelle méthode, l'expertise du biologiste est absolument indispensable pour que les données soient prises en compte dans toute leur complexité. Le statisticien doit donc faire l'effort de comprendre au mieux les données et surtout ne pas hésiter à faire des allers-retours permanents avec les biologistes pour vérifier ses choix. En outre, cette collaboration ne s'arrête pas là. Une fois que le statisticien a mis au point sa méthode, qu'il l'a appliquée à un jeu de données réelles, il est bien incapable d'interpréter les résultats en termes biologiques. En dernier lieu, c'est donc le biologiste qui pourra valider, ou non, les résultats.
Une dernière remarque concernant l'interaction biologiste/statisticien. Il est absolument primordial qui chacun n'oublie pas sa place. Il pourrait être facile, pour un statisticien convaincu par son biologiste préféré, de rechercher, à tout prix, une méthode qui pourrait *faire sortir* ce que le biologiste attend. Or, on sait bien qu'avec ce type de données, quand on sait ce que l'on veut trouver, on peut (presque) toujours arriver à le faire apparaître. Il est donc du devoir du statisticien de mettre en garde le biologiste et de lui dire que malgré le temps et l'argent qu'il a investi dans ces données, il ne peut lui garantir que ces données pourront donner des résultats intéressants.
Toutes ces considérations me semblent être un préalable que l'on doit avoir en tête avant d'initier une collaboration avec une équipe de biologistes. Nous espérons que ces développements n'auront pas ennuyé nos lecteurs, mais il nous semble difficile, pour un statisticien qui souhaite traiter des données réelles, de ne pas s'arrêter pour réfléchir un instant à la difficulté mais surtout à la richesse d'une interaction entre les disciplines si différentes, si passionnantes et si complémentaires que sont la biologie et les statistiques.
En ce qui nous concerne nos principaux partenaires ont été :

- l'équipe CHU-CNRS du Pr Sylvain Lehmann à Montpellier,
- l'équipe Qualité des Produits Animaux, Institut National de la Recherche Agronomique, Centre de Theix à Clermont-Ferrand
- le centre de recherche de la société Altadis.

Les algorithmes génétiques : de la biologie. . . à la biologie

Malgré les apparences, nous venons bien de rentrer dans le domaine des statistiques. Les algorithmes génétiques (notés AG dans la suite) peuvent être classés dans la grande famille d'algorithmes d'optimisation que sont les métaheuristiques et que nous étudierons plus précisément dans la suite. Mais voyons tout de suite l'histoire de ces méthodes.
Dans les années 1960, les avancées dans le domaine de la génétique permettaient de donner naissance au néodarwinisme ou *Théorie synthétique de l'évolution*. La structure de l'ADN avait été publiée en 1953 par Watson & Crick (1953) et avait permis de donner un cadre rigoureux à la théorie de Darwin (Darwin, 1859). La communauté scientifique a mis un certain temps à accepter toutes les implications de telles découvertes. En effet, le néodarwinisme a permis de montrer que l'évolution des espèces au cours de l'histoire de la Terre résulte principalement de la combinaison d'une génération totalement aléatoire de diversité et d'une sélection. Cette diversité provient des mutations de l'ADN et des mécanismes de la reproduction sexuée. Le contraste entre la relative simplicité des mécanismes de base et la complexité des êtres vivants actuels ont démontré l'incroyable efficacité de l'évolution.
Cette efficacité n'a évidemment pas échappé aux scientifiques d'autres domaines. En Allemagne, Rechenberg (1973) et Schwefel (1977) développèrent le concept d'*evolution strategy* tandis qu'aux Etats-Unis, Lawrence Fogel (Fogel, 1963; Fogel *et al.*, 1966) et quelques autres introduisaient le *genetic programming*. Toutes ces contributions avaient en commun l'utilisation conjointe des concepts de mutation et de sélection. Ces deux concepts étaient considérés par Darwin comme prépondérants dans l'évolution naturelle. Cependant, dès 1962, Bremermann et Fraser avaient inclus le concept de croisement dont les scientifiques contemporains avaient mesuré l'importance (bien que Darwin l'ait plutôt négligé). Une bonne partie de ce travail préliminaire a été rassemblé par le fils d'un de ces précurseurs dans Fogel (1998).
Mais, c'est finalement Holland (1975) qui, le premier, a introduit le terme d'*Algorithme génétique* (AG). Dans son travail, la notion de croisement était primordiale. Cependant, il est assez étonnant d'observer que son ouvrage n'était absolument pas centré sur les problèmes d'optimisation. C'est son étudiant qui, dans sa thèse (De Jong, 1975), a focalisé son travail sur l'utilisation des AG dans la résolution des problèmes d'optimisation (De Jong, 1992).
Depuis, les AG ont beaucoup évolué et l'immense majorité des applications concernent des problèmes d'optimisation. Ce développement est essentiellement lié à l'augmentation rapide des capacités de calcul des ordinateurs. En effet, comme nous le verrons par la suite, ces algorithmes sont très *gourmands* en calculs et à l'époque de leur invention, cela limitait beaucoup leurs possibilités d'application. Mais aujourd'hui, les AG se révèlent être des outils d'optimisation très puissants, même pour des problèmes complexes. Or, nous verrons que les problèmes d'optimisation liés aux données de protéomique rentrent tout à fait dans cette catégorie.
Cependant, nous voyons un autre intérêt à l'utilisation des AG pour les données de protéomique, et si cet intérêt est moins technique que le précédent, il n'en est pas moins important pour le succès et l'utilisation des méthodes qui pourraient être mises au point. En effet, nous l'avons vu plus tôt, un traitement efficace des données de protéomique ne pourrait se faire sans une étroite collaboration entre biologistes et statisticiens. Or, il y a peu de chances pour qu'un biologiste soit vraiment enclin à utiliser une méthode statistique s'il la considère comme une boîte noire dont il ne maîtrise ni le fonctionnement, ni les paramètres. C'est pourquoi l'usage des AG dans ce contexte est particulièrement favorable. Le langage nécessaire à la présentation de ces méthodes est tout de suite compris par les biologistes qui intègrent bien

plus facilement les concepts parfois un peu ardus liés à ces méthodes. Chaque étape, chaque paramètre de l'algorithme peut faire l'objet d'une analogie biologique. Cette méthode a donc beaucoup de chance d'être rapidement *adoptée* par une équipe de biologistes même s'ils ne maîtrisent pas tous les développements théoriques que l'on peut faire autour des AG.

Les algorithmes génétiques : un outil puissant aux fondements mathématiques solides

Nous venons de voir l'avantage des AG pour leur adoption par la communauté des biologistes. Mais, ce qui fait cette force devient plutôt un handicap quand on se transpose dans l'univers des statisticiens. En effet, tout comme leur pendant biologique, les AG ont suscité le scepticisme dans la communauté scientifique. Aujourd'hui encore, beaucoup de statisticiens considèrent les AG comme une boîte noire aux pouvoirs magiques dont on est incapable de théoriser le fonctionnement et à laquelle il serait donc illusoire, voire dangereux, de se fier.
Certes, les articles montrant le succès de l'application des AG à des problèmes d'optimisation de plus en plus complexes se sont multipliés. Mais il était clair que, tant que ces méthodes n'auraient pas donné la clef de leur fonctionnement, les statisticiens *sérieux* se refuseraient à les utiliser. Bien que parfois limitant, un tel raisonnement a eu l'immense avantage de pousser les scientifiques à chercher à comprendre ce qui se passe réellement dans un algorithme génétique du point de vue théorique.
Cette recherche a été difficile mais aujourd'hui, elle a rencontré un réel succès. Nous verrons dans la suite de ce travail que bien que les AG soient issus d'une transposition intuitive d'un concept à l'efficacité démontrée, ils s'appuient maintenant, en outre, sur une modélisation qui les rend beaucoup moins mystérieux. Cependant, ces résultats ne sont pas encore suffisamment connus et les préjugés sur les AG ont la vie dure. Nous espérons que ce travail participera à la diffusion de ces résultats et ainsi qu'à leur extension.

Déroulement de ce travail

La première partie de ce travail sera consacrée à un travail de présentation des AG. Elle repose essentiellement sur des recherches bibliographiques mais nous avons tenu à en faire une synthèse qui constitue une introduction relativement complète aux AG. Dans cette partie, nous détaillerons tout d'abord les différentes étapes de la mise en place d'un AG. Il existe de nombreuses variantes dans l'application de ces étapes, nous avons répertorié les plus courantes et donné, quand cela était possible, les avantages et inconvénients de chacune d'elles. Le paragraphe suivant est consacré à la revue de ce qui existe déjà comme grands types de résultats pour la modélisation des AG et l'étude de leur convergence. Enfin, les principaux membres de la famille des métaheuristiques seront présentés en parallèle. Les notations ont été volontairement homogénéisées pour l'ensemble de ces méthodes afin de pouvoir en réaliser une comparaison. Pour cela, nous avons essayé de mettre en place une nomenclature commune qui nous permettra de faire ressortir les principales caractéristiques des différentes méthodes, leurs originalités, leurs points communs et ainsi de voir si l'on peut faire certains choix.

La partie suivante constitue le cœur des résultats sur l'application des AG aux données de protéomique. Ce développement permettra de montrer que, même si les AG sont des méthodes bien codifiées, leur mise en application ne doit jamais être directe et nécessite beaucoup de réflexions sur le problème d'optimisation dans son contexte. Notamment, la construction de la fonction à optimiser requiert beaucoup d'attention.
Nous étudierons tout d'abord la recherche de biomarqueurs en spectrométrie de masse SELDI. Cette technique est utilisée pour repérer les différentes protéines d'un échantillon qui seront séparées en fonction de leur poids et de leur charge électrique. Pour réaliser cette application, nous avons dû développer une méthode de traitement du signal : de l'extraction des pics à leur alignement entre différents spectres. Dans ce contexte, nous avons surtout développé une méthode de discrimination originale qui prend en compte l'ensemble des caractéristiques des spectres de masse SELDI, la *forêt de branches*, ce qui n'est très généralement pas le cas dans les méthodes appliquées par les biologistes à ce type de données. En effet, la méthode prend en compte : la grande variabilité des données (jusqu'à 50 % pour l'ordonnée des spectres obtenus), la nécessité d'utiliser l'information provenant de plusieurs pics dans les spectres, la taille généralement faible des échantillons. La méthode retenue utilise une série d'arbres de décision à un seul nœud utilisés en parallèle. Les paramètres (pics utilisés, seuil choisi) sont optimisés par l'AG. Cette méthode sera appliquée à un jeu de données classique de la littérature ainsi qu'à un jeu de données original. L'utilisation des AG en spectrométrie est également illustrée sur des données différentes issues de la spectrométrie proche infra rouge. Les méthodes développées dans cette partie ont donné lieu à la publication de trois articles. L'autre grande technique d'analyse des données de protéomique est l'utilisation des gels d'électrophorèse en deux dimensions. Un des principaux problèmes de cette technique est sa difficulté de reproductibilité. Pour deux gels obtenus dans des conditions similaires, on observe toujours des distorsions qui empêchent la superposition directe et donc la comparaison des résultats. Nous avons mis au point une nouvelle méthode d'alignement des gels. Cette méthode a pour but de réaliser une série de transformations locales (aucune transformation globale ne serait assez précise). Le choix des zones utilise un partitionnement de Voronoï qui repose sur le choix d'un certain nombre de points de repères (facilement déterminés). Contrairement à la grande majorité des méthodes d'alignement utilisées, notre méthode n'utilise pas un gel de référence auquel seraient alignés tous les autres gels. Ce type de méthodes est très limitant. Pour dépasser ce problème, nous sommes partis de la méthode Procuste généralisée que nous avons étendue pour l'appliquer à des tableaux de données pour lesquels on ne connaît pas les appariements. La méthode a été testée sur un jeu de données précédemment aligné *à la main* par un biologiste. Cette application sera également l'occasion d'aborder un premier aspect de la convergence des AG : l'observation de l'évolution des paramètres au cours des générations et dans la population finale.

Enfin, la dernière partie de ce travail portera sur la caractéristique des AG la plus intéressante du point de vue théorique mais aussi pratique : la convergence. La première partie aura montré les résultats existants et dans cette partie nous nous concentrerons sur ce que nous avons tenté d'apporter à ce domaine.
Dans un premier paragraphe, nous utiliserons la modélisation par chaîne de Markov des populations successives de l'AG avec élitisme ainsi qu'une réduction de l'espace d'états proposée par Bhandari *et al.* (1996) pour l'étendre au cas non homogène et à un type de mutation différent de ce que l'on considère généralement.
Les démonstrations théoriques ont généralement l'inconvénient de ne pas donner de véritable

idée de la vitesse de convergence. Donc, même si ces résultats sont *rassurants*, ils sont inutilisables en pratique. Par ailleurs, d'un point de vue pratique, les critères de convergence utilisés pour arrêter les AG sont très insatisfaisants : nombre de générations maximum atteint, temps de calcul maximum atteint, plus d'évolution significative de la fonction optimisée,... Dans tous les cas, de nombreux réglages sont nécessaires et les critères doivent être adaptés pour chaque nouvelle application. En outre, ces critères sont purement empiriques et ne reposent sur aucun fondement théorique. Le critère que nous avons construit a pour but de réaliser un compromis entre les développements théoriques inapplicables en pratique et les critères *à vue de nez* généralement employés. Pour cela, nous avons construit une nouvelle chaîne de Markov qui compte, dans les dernières populations, le nombre d'occurrences de la solution localement optimale. L'observation à l'origine de ce critère est le fait qu'une solution réellement optimale va pouvoir *envahir* la population. Nous verrons que ce critère ne peut garantir d'obtenir l'optimum global (c'est le prix à payer pour avoir un critère applicable en pratique !) mais il repose sur une modélisation réelle du fonctionnement de l'AG qui pourrait donner des résultats intéressants. Ce critère a été testé sur différents AG que nous avons mis au point par ailleurs. Les conditions d'applicabilité de ce critère sont également étudiées.
Le dernier paragraphe repose sur une dernière observation qui nous a montré l'influence de l'envahissement de la population par un sous-optimum qui peut empêcher une bonne émergence d'un meilleur optimum et donc empêcher d'utiliser correctement le critère de convergence dont nous venons de parler. Pour résoudre ce problème, nous sommes retournés à la biologie et nous avons construit un AG qui mime le phénomène de catastrophe naturelle et d'épisodes de grandes extinctions qui ont parsemé l'histoire de l'évolution des espèces. Cette méthode sera appliquée à un problème de classification non supervisée.

Première partie

Présentation des algorithmes génétiques

Chapitre 1

De l'évolution des espèces à l'optimisation mathématique : étapes de l'algorithme génétique

Comme nous l'avons développé dans l'introduction, les origines des AG sont assez intuitives. Mais nous allons voir qu'ils impliquent des aspects assez techniques. En effet, dans cette partie, nous allons tout d'abord détailler les différentes étapes de la mise en place d'un algorithme génétique : nous étudierons le codage, l'initialisation et les opérateurs de mutation, de croisement et de sélection. Nous présenterons ensuite les résultats existants sur la modélisation des AG. Finalement, nous considérerons quelques représentants majeurs de la famille dont font partie les AG : les métaheuristiques.

Bien qu'elle n'était pas l'objectif premier des AG, l'optimisation représente aujourd'hui l'écrasante majorité des applications des AG. C'est cet usage que nous allons considérer. Dans cette partie, nous allons considérer comme problème d'optimisation, la maximisation d'une fonction objectif f (l'espace $\mathcal{X}$ sera défini en détail dans les paragraphes suivants) :

$$f : \mathcal{X} \to \mathbb{R}.$$

Il va de soi que les adaptations permettant la minimisation d'une telle fonction est immédiat.

Illustration
Afin de mieux comprendre la mise en place des différentes étapes de l'AG, nous allons utiliser une application simple. Cette application concerne l'Analyse en Composantes Principales dite *Simple* (Vines, 2000; Rousson & Gasser, 2004) notée SCA (Simple Component Analysis). Cette méthode consiste à rechercher des combinaisons linéaires des variables de départ sous forme d'entiers de sorte à optimiser le critère classique de l'ACP (maximisation de l'inertie de chaque axe). On nomme *axes* le vecteurs contenant les coefficients et *composantes* les vecteurs obtenus en appliquant ces coefficients aux variables de départ : si on note $\mathbf{X}$ la matrice des données de départ, $\mathbf{c}_i = \mathbf{X}\mathbf{a}_i$, où $\mathbf{a}_i$ est le $i^{\text{ème}}$ axe et $\mathbf{c}_i$ la $i^{\text{ème}}$ composante. L'objectif est d'obtenir des composantes dont l'interprétation vis-à-vis des variables de départ est simple. Nous allons montrer dans les paragraphes suivants que cette problématique est tout à fait adaptée à une résolution par AG, ce qui conduit à une méthode que nous avons appelée GA-SCA.

1.1 Le codage : du problème biologique à l'optimisation mathématique

Une des grandes avancées du néodarwinisme a été le décryptage du code génétique. Cette découverte a révélé une chose extrêmement surprenante : toute la complexité du monde vivant était codée par un alphabet de quatre lettres (A, C, T et G) utilisé pour construire des mots de trois lettres seulement (chaque mot code pour un acide aminé, constituant élémentaire des protéines). Notons que ce code, qui est le plus simple, est loin d'être le seul dans le monde vivant. La première difficulté d'un AG est également de réaliser un codage des solutions, c'est-à-dire de passer d'une solution dans le domaine d'application (la biologie dans notre cas) à une solution dans le domaine des AG.
Afin de pouvoir appliquer les opérateurs, chaque solution doit être complètement définie par un vecteur numérique. Un premier travail doit donc être réalisé, en collaboration avec les experts du domaine, pour extraire des solutions du domaine d'application ce qui doit être utilisé pour caractériser les solutions dans l'espace des solutions des AG.
Pour construire l'espace des solutions des AG, on a besoin de se munir d'un alphabet $\mathcal{A}$. Chaque solution pourra alors être représentée par un vecteur $\mathbf{x} \in \mathcal{A}^l$ où l est la longueur nécessaire au codage. Cette longueur dépend de card($\mathcal{A}$) et de l'espace d'application. En effet, plus les solutions dans l'espace d'application sont complexes (beaucoup d'information à prendre en compte), plus la longueur nécessaire au codage sera importante. De même, pour une même complexité des solutions, moins l'alphabet comportera d'éléments, plus l devra être grand pour coder la même information.
En outre, l'espace d'application peut impliquer des contraintes qui font que tous les vecteurs de $\mathcal{A}^l$ ne représentent pas des solutions *valables*. On note alors $\mathcal{X} \subseteq \mathcal{A}^l$ l'espace des solutions de l'AG restreint aux solutions valables.

Historiquement, le codage choisi était le codage binaire. Tout d'abord, l'analogie avec la structure de l'ADN et le codage en quatre bases est immédiate. De plus, l'application des opérateurs est très simple car un 1 ne peut devenir qu'un 0 et inversement. Holland considérait en fait que le codage binaire était optimal. Il a énoncé cela sous la forme du *Principe de l'Alphabet Minimum* qui indique globalement que l'on a plus de possibilités d'encodage avec un alphabet de cardinalité minimale, c'est-à-dire de cardinalité deux, donc un codage binaire. La démonstration de ce principe repose sur la théorie des schémas (Holland, 1975). Cependant, aujourd'hui cet argument a été invalidé, notamment par Antonisse (1989) qui a montré que la notion de schéma a été spécialement pensée par Holland pour le codage binaire et que sa signification pour un autre codage doit être revue. Ceci rend abusive toute conclusion découlant de la théorie des schémas de Holland pour un codage autre que binaire.
On peut considérer que l'usage des AG sous forme binaire est plus simple, notamment en ce qui concerne la modélisation. Pendant longtemps en effet, les seuls résultats théoriques disponibles sur les AG ne concernaient que le codage binaire (Cerf, 1994). De plus, pour ce qui est de l'estimation d'une taille de population minimale, une des conditions possibles est que chaque point de l'espace des solutions doit pouvoir être atteint uniquement par croisement des individus de la population initiale. Pour cela, il faut qu'à chacune des l positions (appelées *loci* dans Holland (1975)), toutes les valeurs possibles soient présentes dans la population initiale. Si on a un codage binaire, il n'y a que deux valeurs possibles par locus ce

qui réduit considérablement la taille de la population nécessaire pour satisfaire ce principe. Il y a donc un certain nombre d'arguments en faveur du codage binaire.
Cependant, il existe maintenant des résultats théoriques qui s'appliquent quel que soit card($\mathcal{A}$) (Bhandari *et al.*, 1996). De plus, le codage par valeurs réelles ou entières, par exemple, permet une longueur de codage l bien plus réduite et le temps de calcul peut alors diminuer significativement. En outre, le codage binaire impose, pour la plupart des problèmes, une sorte de *deuxième couche* de codage. En effet, il faut tout d'abord traduire un problème dans le domaine d'application en un vecteur numérique mais ce vecteur numérique est rarement directement binaire. Il faut donc, la plupart du temps, recoder les variables obtenues sous forme binaire. Si on a beaucoup de variables à coder et que l'on veut une certaine précision, on obtient rapidement des longueurs de codage l très importantes.
Toutefois, le choix du codage reste un problème ouvert. Certains articles ont montré une suprématie empirique du codage binaire (Rees & Koehler, 1999) alors que d'autres (Janikow & Michalewicz, 1991) ont montré le contraire. Il semble que la pertinence du choix du codage dépend fortement du problème, spécialement de la complexité des solutions à coder et de la précision désirée.
Dans les applications qui suivront, nous avons choisi d'utiliser un codage direct, sans passer par le codage binaire.

Illustration
Pour ce qui concerne GA-SCA, une solution dans l'espace d'application est représentée par p entiers qui correspondent aux coefficients appliqués aux p variables de départ pour obtenir les composantes. Le codage est donc très simple dans ce cas, une solution consistera en un vecteur de p entiers. On a donc ici, $\mathcal{A} = \mathbb{Z}$. Avec ce codage, on aura $l = p$ alors que si on avait fait un codage binaire, cette longueur aurait été bien supérieure.

1.2 L'initialisation : une étape dépendante du problème

L'objectif de l'étape d'initialisation est de choisir un ensemble de solutions potentielles au problème d'optimisation posé. En fait, chaque solution potentielle va représenter un *individu* (dit aussi *chromosome* dans la terminologie de Holland (1975)). Tous ces individus vont être rassemblés dans ce que l'on nommera la *population initiale*.
Deux questions principales se posent lors de la construction de cette population initiale :
- Quel doit être le nombre T_{pop} d'individus dans la population initiale ?
- Comment doit-on générer les solutions initiales ?

Pour ce qui est du nombre d'individus (ou taille de la population), on peut à nouveau faire l'analogie avec l'évolution des espèces, il a été observé qu'une petite population peut évoluer beaucoup plus vite qu'une population plus importante. En effet, si un caractère favorable est présent chez un ou plusieurs individus de la population, ce caractère pourra rapidement se propager (par la reproduction) dans une population de taille réduite alors qu'il faudra beaucoup de temps pour qu'il se répande dans une grande population. Cependant, une population de taille importante est un réservoir de plus grande diversité génétique qui permet une possibilité d'adaptation à une plus grande diversité de situations environnementales.
Dans le cas des AG, nous devons également réaliser un compromis entre deux objectifs contradictoires : minimiser le temps de calcul et limiter le risque d'obtenir un optimum local. Concernant le temps de calcul, il dépend évidemment du nombre d'individus dans

la population. En effet, comme nous le verrons par la suite, chaque opérateur (mutation, croisement et sélection) s'applique à un individu ou à un petit groupe d'individus. Ainsi, plus on aura d'individus dans la population, plus il faudra appliquer les opérateurs et donc augmenter le temps de calcul. Quant au problème des optima locaux, plus la population est grande, plus on a de chances de bien explorer l'espace des solutions et moins on a de chances d'avoir de grandes zones de cet espace inexplorées. On doit donc réaliser un équilibre entre ces deux objectifs.
Il est assez évident que, plus la longueur d'un individu, l, sera grande plus il faudra que la taille de la population soit importante pour assurer une même qualité d'exploration de l'espace des solutions. Certains se sont intéressés à une détermination de la dépendance entre taille *optimale* de la population et longueur des individus.
Par la théorie des schémas, Goldberg (1985, 1989a) a conclu, globalement, que la taille de la population devait suivre une fonction exponentielle de la longueur des individus... ce qui peut rapidement conduire à des tailles de population très importantes. En utilisant un autre point de vue, Goldberg *et al.* (1992) ont finalement déduit qu'une dépendance linéaire suffisait, ce qui, dans certains cas, peut déjà se traduire par de grandes tailles.
Nous avons vu dans la partie 1.1 que certains utilisaient un critère bien précis pour décider de la taille minimale de la taille de la population : il faut que tous les points de l'espace des solutions puissent être atteints en utilisant uniquement le croisement sur les solutions de la population initiale. Cela impose que toutes les valeurs possibles (nommées *allèles* par Holland (1975)), pour chaque locus, soient présentes dans la population initiale. Pour un codage binaire, cette taille minimale peut être déterminée par (Reeves & Rowe, 2003) :

$$N \approx \lceil 1 + log(-l/lnP_1)/log2 \rceil ,$$

où P_1 est la probabilité souhaitée de trouver au moins une fois chaque allèle pour chaque locus. Par exemple, pour $l = 50$ et $P_1 = 0.999$, on obtient $N = 17$.
Suivant ce principe, il est évident que plus la cardinalité de l'alphabet est importante plus la taille de la population doit être grande. Cependant, ce principe ne tient pas du tout compte de l'effet de la mutation qui, on le verra dans la suite, est primordial.

Le deuxième problème de la construction de la population initiale est la méthode de génération des individus. D'un point de vue général, on peut dire qu'une hétérogénéité maximale est souhaitable (analogue avec la diversité génétique des populations). En effet, plus la population initiale sera hétérogène, plus la proportion de l'espace des solutions explorée sera importante et donc, moins on aura de risque de manquer l'optimum global. Cette hétérogénéité peut être générée aléatoirement ou de façon plus déterministe.
Pour la méthode aléatoire, cela consiste à choisir chaque allèle initial dans $\mathcal{A}$. Ce choix peut être fait de manière uniforme ou en favorisant certains allèles si l'on dispose d'une information préalable. Il convient naturellement de s'assurer de la qualité du générateur de séquences pseudo-aléatoires utilisé. Plus card($\mathcal{A}$) est important plus le générateur doit être performant.
On peut également utiliser une méthode plus systématique de génération d'hétérogénéité, notamment si l'on veut se rapprocher du principe de présence de tous les allèles pour chaque locus dans la population de départ. Par exemple, il est possible d'utiliser la généralisation de l'hypercube latin (McKay *et al.*, 1979).
Le principe d'hétérogénéité de la population initiale est tout à fait recommandé dans la plupart des cas, on peut même dire qu'il est possible dans tous les cas. Cependant, quand on a des informations *a priori* sur les solutions à obtenir, on peut tout à fait utiliser cette

information pour accélérer la convergence de l'algorithme et donc réduire le temps de calcul. Cette technique est à double tranchant car si l'information *a priori* est mauvaise, on peut converger rapidement vers un optimum local et manquer l'optimum global. Cette méthode doit donc être utilisée avec précaution.
Enfin, on peut décider de faire un mélange des deux techniques en générant de façon tout à fait aléatoire une partie de la population mais en introduisant aussi des solutions *a priori* bonnes. L'information *a priori* peut alors être utilisée de différentes façons : on peut introduire directement les solutions identifiées ou générer des solutions dans leur voisinage (soit en fixant certains loci et en laissant les autres libres, soit en définissant un voisinage autour de chaque locus).
En conclusion, la génération aléatoire peut être utilisée dans toutes les situations et doit être préférée quand on n'a pas d'information *a priori* fiable. Si toutefois une telle information est disponible, il peut être judicieux de s'en servir afin d'accélérer la convergence de l'AG.

Illustration

Revenons au cas de GA-SCA. Nous ne disposons pas d'information *a priori* sur les valeurs des coefficients entiers à appliquer aux variables de départ. Cependant, notre objectif est d'obtenir des résultats interprétables. On a donc tout à fait intérêt à avoir des coefficients assez petits. Par exemple, une solution qui contiendrait des zéros, des uns et des deux serait tout à fait simple à interpréter : certaines variables ne joueraient aucun rôle (coefficient zéro) et d'autres variables (coefficient deux) auraient deux fois plus d'influence que les dernières (coefficient un). Mais on pourrait tout aussi bien coder la même information avec des zéros, des 1989 et des 3978. Cependant, toute l'information interprétable peut être codée avec des entiers petits (en valeur absolue). En effet, qu'une variable ait un coefficient mille fois ou 1400 fois plus grand qu'une autre ne modifie pas significativement l'interprétation. On peut donc coder les rapports d'influence des variables en utilisant des entiers relatifs assez petits. Afin de se rapprocher de telles solutions, nous avons choisi d'initialiser les T_{pop} solutions avec le même vecteur de p uns. Certes, ce type d'initialisation ne permet pas une bonne exploration de l'espace des solutions mais il nous semble que cette méthode porte un réel sens. Ainsi, au départ, chaque variable joue le même rôle dans la construction des axes, c'est le cas le moins informatif, et petit à petit, on fait varier ce rôle. Alors, chaque changement sur un des coefficients aura des conséquences importantes sur la qualité de la solution obtenue alors que si on change un 1000 en 1001, les solutions obtenues seront très proches. De plus, dans l'ensemble des exemples que nous avons traités, nous n'avons jamais obtenu de très gros coefficients et ces coefficients sont généralement centrés autour de 0. Il semble donc que partir d'un vecteur de uns constitue une initialisation qui limite le chemin vers n'importe quelle solution de ce type. Il serait évidemment possible de partir d'une initialisation plus hétérogène mais dans les exemples que nous avons traités, la méthode décrite ci-dessus a été efficace.
On voit bien ici, l'importance de la maîtrise du problème de départ pour construire des solutions initiales pertinentes et les plus à même d'évoluer vers des solutions intéressantes.

1.3 L'opérateur de mutation : l'explorateur de l'espace des solutions

Le terme même de mutation vient directement de la biologie. Il désigne des changements dans la séquence de l'ADN des individus. Cette mutation peut être due à des facteurs extérieurs (radiations, produits chimiques,...) ou résulter d'erreurs non corrigées apparues lors de la duplication de l'ADN. Ces mutations ont des conséquences plus ou moins importantes suivant les organes qu'elles atteignent (une mutation affectant un gamète pourra être transmise aux descendants de l'individu, ce qui n'est pas le cas pour une autre cellule) et les conséquences sur les protéines issues de la traduction des séquences mutées. Certaines n'ont aucune conséquence, d'autres conduisent à des protéines non fonctionnelles et enfin certaines induisent la construction de protéines aux nouvelles fonctions. Ce sont ces dernières qui sont les plus intéressantes car elles peuvent permettre l'apparition de nouvelles adaptations chez les individus. Notons également l'existence de mutations portant sur des zones du génome qui régulent la transcription dont les conséquences peuvent être tout aussi importantes. Ce qu'il faut bien garder à l'esprit, c'est que ces mutations sont tout à fait aléatoires et résultent de phénomènes qui tiennent de l'erreur. Il faut donc réaliser de nombreuses mutations pour en obtenir une seule qui aura des conséquences favorables.
Dans le domaine des AG, nous allons tenter de reproduire le même genre de mécanisme. Dans notre cas, l'opérateur de mutation permet d'introduire l'aléa nécessaire à une exploration efficace de l'espace des solutions. En effet, il va permettre d'introduire des valeurs pour les allèles qui n'étaient pas forcément présentes à un locus dans la population de départ. On voit donc qu'en utilisant la mutation, il n'est pas nécessaire de construire une population initiale contenant tous les allèles à tous les loci. De plus, l'opérateur de mutation va permettre, tout au long de l'algorithme de maintenir une certaine homogénéité dans la population et ainsi, d'éviter une convergence trop rapide vers un optimum local.
Les mutations ne s'appliquent pas à tous les individus de la population à chaque génération. On définit un taux de mutation qui indique quelle proportion moyenne de la population doit subir une mutation. Le taux de mutation peut s'appliquer à différents niveaux. En effet, on peut appliquer le taux de mutation à chaque locus ou globalement, pour chaque individu. De même, le taux de mutation peut être constant et défini pour tout l'algorithme mais on peut aussi décider de le faire varier. Dans le cas d'un taux de mutation non constant, on peut le faire dépendre du locus (Fogarty, 1989), de la diversité de la population (mesurée par l'écart-type de la valeur de la fonction à optimiser dans la population) (Reeves, 1995). On peut aussi faire varier le taux de mutation dans le temps (Pittman, 2002) afin de contrôler l'hétérogénéité dans la population. En effet, au début de l'algorithme, on a tout intérêt à favoriser une bonne exploration de l'espace des solutions et donc à avoir un taux de mutation élevé. Ensuite, pour permettre à l'AG de converger, on peut diminuer le taux de mutation. Enfin, pour permettre à l'algorithme de sortir d'un éventuel optimum local, on peut appliquer un taux de mutation plus élevé vers la fin de l'algorithme. La gestion de ces paramètres concernant le taux de mutation doit être adaptée au problème d'optimisation.
Pour ce qui est de l'application de l'opérateur de mutation, elle dépend du codage adopté. En effet, si le codage binaire est utilisé, le un ne peut devenir qu'un zéro et réciproquement. Alors que si on utilise un codage avec $\text{card}(\mathcal{A}) \geq 3$, des choix doivent être faits. Globalement, si on a décidé de le muter, on peut autoriser un locus à prendre uniformément toutes les valeurs possibles de $\mathcal{A}$ ou imposer une structure de voisinage autour de l'ancienne valeur.

L'utilisation de mutations complètement libres permet une meilleure exploration de l'espace des solutions, cependant, comme pour l'initialisation de la population, si on dispose d'une information *a priori* concernant le problème d'optimisation, on peut restreindre les possibilités afin d'obtenir une convergence plus rapide. Dans ce cas, on a toujours le risque, si l'information n'est pas pertinente, de guider l'AG vers un optimum local.
Il est très important de garder à l'esprit que l'opérateur de mutation agit de façon tout à fait aléatoire, tant pour le choix des individus et des loci qui subissent la mutation que pour le choix de la nouvelle valeur. On n'a donc aucune garantie concernant la qualité de la solution obtenue, on peut tout aussi bien avoir dégradé qu'amélioré la solution initiale par rapport à la solution à optimiser.

Illustration
Nous avons vu dans le paragraphe 1.1 que, pour GA-SCA, on avait $\mathcal{A} = \mathbb{Z}$. On est donc dans le cas où l'on doit faire un choix quant à la structure de voisinage autorisée pour la mutation d'un locus (card($\mathcal{A}$)). Or, d'après le paragraphe 1.2, nous savons que la population de départ est initialisée avec des vecteurs composés de uns et que, dans ce cas, le moindre changement de valeur d'un locus a d'importantes répercussions sur les solutions obtenues. Il a donc été décidé de n'autoriser que deux mutations pour un locus : si l'allèle présent sur le locus est z alors, après mutation, il prendra soit la valeur $z + 1$, soit la valeur $z - 1$ avec équiprobabilité. La solution obtenue remplace la solution de départ.
Pour ce qui de la définition du taux de mutation, chaque locus d'une solution a le même sens que les autres, le taux de mutation s'applique donc uniformément, sur chaque locus. Dans cette application, nous avons choisi un taux de mutation constant au cours des générations.

1.4 L'opérateur de croisement : le catalyseur

Lors de la présentation de l'émergence des AG, nous avons indiqué que l'opérateur de croisement (*cross over* en anglais) était apparu, dans les précurseurs des AG, postérieurement à l'opérateur de mutation. Cela suivait tout à fait la progression de la pensée chez les évolutionnistes. Dans Holland (1975), le croisement occupait au contraire la place centrale et, de nos jours, chez certains auteurs, l'opérateur de mutation est considéré comme secondaire. Cependant, quand on raisonne en termes de modélisation et de convergence des AG, ce raisonnement semble difficile à tenir.
L'objectif de l'étape de croisement est de combiner les allèles de plusieurs individus. Nous verrons dans la suite que, grâce à l'opérateur de sélection, les solutions de la population sont censées être de plus en plus performantes au sens du problème d'optimisation. On peut donc espérer, en combinant les caractéristiques de solutions performantes, obtenir des solutions encore plus performantes.
Il va de soi que, tout comme pour les mutations, les conséquences de l'opérateur de croisement sont aléatoires. On peut tout à fait, par l'échange d'allèles de solutions très performantes, générer des solutions médiocres. En effet, il existe, la plupart du temps, une interaction entre les variables codées dans le calcul de la fonction à optimiser, alors, changer les associations entre allèles peut détruire cette interaction.
C'est exactement ce qui se passe dans la nature, pour toutes les espèces se reproduisant de façon sexuée. Dans la reproduction sexuée, chacun des parents transmet la moitié de son génome à sa descendance. Cette moitié est choisie de façon aléatoire lors de la fabrication

des gamètes. Différents mécanismes interviennent dans ce choix (dont un appelé *crossover*) de sorte à ce qu'il y ait un nombre extrêmement important de possibilités. Choisir la reproduction sexuée, c'est prendre le risque de perdre une bonne partie d'un génome performant, puisque porté par un individu qui a été capable de se reproduire. C'est donc un pari sur l'avenir. En effet, dans le monde vivant terrestre, les espèces les plus *évoluées* ou du moins complexes sont celles qui pratiquent la reproduction sexuée et qui se donnent la possibilité de faire apparaître de nouvelles combinaisons de gènes. La reproduction sexuée est un grand générateur de diversité donc de possibilités d'adaptation.
Dans le domaine des AG, l'application concrète de l'opérateur de croisement est bien moins simple que pour l'opérateur de mutation (cette partie est inspirée de Reeves & Rowe (2003)). De nombreuses options sont possibles. Il faut tout d'abord choisir combien d'individus (les *parents*) seront croisés et combien d'individus (les *enfants*) résulteront de ce croisement. Dans ce travail, nous allons uniquement considérer le cas, très courant dans les AG où deux parents engendrent deux enfants. Cette méthode permet de conserver une taille de population constante, si on remplace les parents par les enfants. Ensuite, on doit décider des positions pour lesquelles on va échanger les allèles des deux parents.

Le cas le plus élémentaire est le *croisement à un point*. Cela consiste simplement à choisir un locus parmi les l possibles comme point de croisement et à intervertir entre les parents les allèles des loci qui se trouvent après le point de croisement. Prenons un exemple et considérons les deux parents suivants, P_1 et P_2, pour $l = 6$:

$$P_1 = (a_1, a_2, a_3, a_4, a_5, a_6) \quad \times \quad P_2 = (b_1, b_2, b_3, b_4, b_5, b_6).$$

Si on choisit comme point de croisement le quatrième locus, on obtient les enfants suivants, E_1 et E_2 :

$$E_1 = (a_1, a_2, a_3, a_4, b_5, b_6) \quad \text{et} \quad E_2 = (b_1, b_2, b_3, b_4, a_5, a_6).$$

Comme souligné par Eshelman *et al.* (1989), le problème majeur du croisement à un point, c'est qu'il permet peu la dissociation des loci contigus. C'est ce que Eshelman *et al.* (1989) appellent le phénomène du *biais positionnel*. Les loci voisins ont tendance à rester ensemble. Or, il n'y a aucune raison pour que cela soit favorable à l'optimisation. Le croisement à un point ne semble donc pas être la solution idéale.

On peut également envisager un croisement à points multiples : il s'agirait de générer plusieurs points de croisement et d'intervertir les fragments obtenus. Si on a suffisamment de points de croisement, le biais positionnel sera réduit. Le *croisement uniforme* représente une sorte de généralisation du croisement à points multiples. Il consiste à générer un vecteur $\mathbf{c}$ de longueur l. A chaque position de ce vecteur, on affecte aléatoirement une valeur *zéro* ou *un* par une distribution de Bernoulli. Si la valeur du paramètre de cette loi est 0.5, on a la même contribution des deux parents mais on peut tout à fait jouer sur ce paramètre pour favoriser les caractéristiques de l'un des parents (par exemple, en fonction de leur valeur relative, par rapport à la fonction à optimiser). On construit un vecteur $\overline{\mathbf{c}}$ qui est le complément de $\mathbf{c}$. On obtient alors la descendance de la manière suivante :

$$E_1 = \mathbf{c} \otimes P_1 \oplus \overline{\mathbf{c}} \otimes P_2 \quad \text{et} \quad E_2 = \overline{\mathbf{c}} \otimes P_1 \oplus \mathbf{c} \otimes P_2,$$

où $\oplus$ et $\otimes$ sont respectivement l'addition et la multiplication élément par élément.

Fig. 1.1 Pseudo-code du croisement à n points généralisé.

On choisit aléatoirement un entier $n \in \{1, 2, \ldots, l-1\}$.
On choisit aléatoirement n points de croisement.
On obtient $n+1$ fragments.
On génère une permutation aléatoire $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{n+1})$ de $(1, 2, \ldots, n+1)$.
On désigne un des parents.
pour $i = 1, \ldots, n+1$ **faire**
    On copie tous les allèles compatibles du fragment σ_k provenant du parent désigné.
    On change de parent désigné.
fin
si l'enfant produit est incomplet **alors**
    On insère les allèles compatibles pour les loci requis.
fin.

Cependant, il existe au moins un cas où aucune des méthodes décrites ci-dessus n'est pas applicable : le cas où un individu représente une permutation (cas du fameux problème du voyageur de commerce). Par exemple, on peut rechercher à optimiser la permutation des entiers de 1 à 7. Dans ce cas, on pourrait avoir deux solutions qui soient $P_1 = (1, 6, 4, 2, 3, 5, 7)$ et $P_2 = (6, 4, 3, 2, 5, 7, 1)$. Si on appliquait à ces parents un croisement à un point où le point de croisement serait la deuxième position, on obtiendrait les enfants, $E_1 = (1, 6, 3, 2, 5, 7, 1)$ et $E_2 = (6, 4, 4, 2, 3, 5, 7)$. Il est évident que l'on n'obtient pas des permutations. En effet, E_1 comporte deux fois le un et E_2, deux fois le quatre. Les permutations ne sont qu'un exemple et de nombreux autres codages se heurtent à ce type de problème. Radcliffe & Surry (1995) ont proposé un formalisme (Fig. 1.1) qui permet d'englober toutes les méthodes de croisement. On peut appeler cela, le croisement à n points généralisé.

Pour l'illustrer reprenons l'exemple des permutations avec $P_1 = (1, 6, 4, 2, 3, 5, 7)$ et $P_2 = (6, 4, 3, 2, 5, 7, 1)$ comme parents. Admettons que l'on ait choisi $n = 2$ avec comme points de croisement, les loci 3 et 5. Chaque parent se retrouve alors divisé en $n+1 = 3$ fragments. On choisit $\sigma = (3, 1, 2)$ et on désigne le parent P_1. Pour $i = 1$, on n'a pas encore inséré d'allèles donc tous sont compatibles (pas de double possible). On copie le fragment $\sigma_1 = 3$ de P_1, c'est-à-dire $(5, 7)$, on obtient :

$$(-, -, -, -, -, 5, 7).$$

Pour $i = 2$, on travaille avec P_2 et le fragment $\sigma_2 = 1$, soit $(6, 4, 3)$, aucun de ces allèles n'est déjà présent dans l'enfant, tout est donc compatible, on obtient :

$$(6, 4, 3, -, -, 5, 7).$$

Enfin, pour $i = 3$, on travaille à nouveau avec P_1 et on choisit le fragment $\sigma_3 = 2$, c'est-à-dire $(2, 3)$. L'allèle 3 est déjà présent dans l'enfant, on ne peut donc pas l'insérer à nouveau, on n'insère que le 2 qui, lui, n'est pas encore présent, on obtient :

$$(6, 4, 3, 2, -, 5, 7).$$

Enfin, on complète avec le seul allèle compatible pour la dernière position, le 1. Finalement, on a :

$$E_1 = (6, 4, 3, 2, 1, 5, 7).$$

Fig. 1.2 Pseudo-code de l'application de l'opérateur de croisement pour p_c fixé.

On génère une permutation aléatoire $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{T_{pop}})$ de $(1, 2, \ldots, T_{pop})$.
pour $i = 1, \ldots, (T_{pop}/2)$ **faire**
    On génère μ uniformément dans $[0, 1]$.
    si $\mu \leq p_c$ **faire**
        On applique le type de croisement choisi avec pour P_1 le $(\sigma_{2i-1})^{\text{ème}}$ individu de la population et pour P_2, le $(\sigma_{2i})^{\text{ème}}$.
    fin
fin

Pour obtenir le deuxième enfant, on peut appliquer les mêmes paramètres mais en changeant l'ordre de désignation des parents. Pour $i = 1$, on extrait le fragment $(7, 1)$ de P_2, on obtient :

$$(-, -, -, -, -, 7, 1).$$

Pour $i = 2$, on extrait le fragment $(1, 6, 4)$ de P_1, mais l'allèle 1 est déjà présent dans l'enfant, on n'insère donc que le 6 et le 4, on obtient :

$$(-, 6, 4, -, -, 7, 1).$$

Pour $i = 3$, on extrait le fragment $(2, 5)$ de P_2, tous les allèles sont à nouveau compatibles, on obtient donc :

$$(-, 6, 4, 2, 5, 7, 1).$$

Il ne reste alors plus qu'à compléter par l'allèle manquant et finalement,

$$E_2 = (3, 6, 4, 2, 5, 7, 1).$$

Cette méthode a l'avantage de pouvoir s'appliquer quel que soit le cas de figure avec, éventuellement, la nécessité d'ajouter de l'aléa quand il est nécessaire de compléter l'enfant obtenu dans la dernière étape. Dans notre exemple, il ne manquait qu'un allèle à la dernière étape, il n'y avait donc qu'une seule possibilité. Mais s'il manque plusieurs allèles, plusieurs positions sont possibles pour chaque allèle et on peut alors les répartir aléatoirement.

Une fois que l'on a choisi le type de croisement à réaliser, il faut, comme dans le cas de la mutation, décider quels individus de la population subiront le croisement. Il est nécessaire de définir un taux de croisement p_c : en moyenne, $T_{pop} \times p_c$ individus de la population subiront le croisement. Pour appliquer l'opérateur de croisement, dans le cas où T_{pop} est pair, nous avons choisi la méthode décrite dans la Fig. 1.2.

Notons à nouveau que le nombre d'enfants obtenus par croisement n'est pas forcément égal au nombre de parents, on peut n'en générer qu'un, ou en choisir un au hasard parmi ceux générés ou encore choisir le meilleur des enfants générés selon un critère comme la valeur du critère à optimiser, la diversité de la population,... on peut alors, ou non, garder un des parents.
Il est aisé de constater, en étudiant le mode d'action de l'opérateur de croisement, que l'effet de son application sur les individus de la population est tout à fait aléatoire du point de vue du critère à optimiser. Comme pour l'opérateur de mutation, il est impossible de prévoir à

l'avance si l'application du croisement va produire des individus plus ou moins performants que les parents dont ils sont issus.
Nous avons indiqué dans le titre de cette partie que nous considérons l'opérateur de croisement comme un catalyseur. En effet, il va permettre, pour des allèles présents, d'accélérer l'optimisation de leur combinaison. Cette propriété sera détaillée dans le paragraphe 1.3.5.

Illustration
Le cas de GA-SCA est particulier du point de vue des croisements car, comme nous l'avons vu, plusieurs chromosomes peuvent désigner exactement la même solution : par exemple, $(1, 0, -1, 1, 2, 1)$ est tout à fait équivalente à $(10, 0, -10, 10, 20, 10)$. Or, si on croise ces deux solutions, on n'obtient pas du tout une solution équivalente. Pour résoudre ce problème, on peut, pour chaque solution, diviser chaque locus par le plus grand dénominateur commun de l'ensemble des loci du chromosome. Dans ce cas, on n'a plus aucun problème pour appliquer l'opérateur et tous les types de croisements décrits dans les paragraphes précédents peuvent être utilisés. Le croisement uniforme, par exemple, semble tout à fait adapté. Après application du croisement, les deux enfants remplacent les parents.

1.5 L'opérateur de sélection : la seule étape *intelligente*

Dans le cadre de la théorie de l'évolution, on parle de *sélection naturelle*. Cette notion recouvre le fait qu'un individu bien adapté à son environnement aura plus de chances de survivre. et surtout de se reproduire, donc de transmettre ses gènes. Mais, dans ce problème d'optimisation, il est assez difficile de définir l'analogue d'une *fonction* optimisée par l'évolution, fonction que Dawkins (1995) appelle *fonction d'utilité de Dieu*. Dès les premières observations, on peut constater que toute idée liée à une quelconque morale doit être définitivement écartée. En fait, il semble à Dawkins et à d'autres évolutionnistes que le but de la vie de façon très générale est certes de survivre, mais surtout d'avoir assez d'atouts pour avoir le temps de se reproduire (ce qui peut sembler une bonne justification de la mortalité des êtres vivants qui n'ont plus d'*utilité* après s'être reproduits). On peut donc penser que, finalement, c'est la survie des gènes qui est maximisée par la sélection naturelle. Une vision assez effrayante qui ne ferait des êtres vivants que des vecteurs temporaires au service de leurs gènes. A méditer...

Revenons aux AG. Nous avons déjà vu que, dans notre contexte, l'objectif de l'AG est l'optimisation d'une fonction d'intérêt. Or, jusqu'à maintenant, les opérateurs que nous avons décrits avaient uniquement un rôle exploratoire sans discrimination des solutions obtenues. C'est l'opérateur de sélection qui va permettre d'évaluer les solutions obtenues après application de la mutation et du croisement, en fonction de leur valeur pour le problème d'optimisation. La première étape pour pouvoir appliquer l'opérateur de sélection est donc de définir la fonction à optimiser, la *fonction d'utilité* de l'AG.

1.5.1 Construction de la *fitness*

Le fonction objectif a deux buts : optimiser la fonction d'intérêt et prendre en compte d'éventuelles contraintes liées au problème d'optimisation. Nous allons donc distinguer la

fonction liée au problème d'optimisation de départ, que nous nommerons fonction objectif et la fonction complète qui sera réellement optimisée par l'AG que l'on appelle *fitness*. La fonction objectif est plus ou moins simple à déterminer, cela dépend du problème posé. Dans certains cas (classification, régression,...), elle est évidente, mais dans d'autres cas (cf. partie II), le choix de la fonction objectif représente la plus importante partie du travail. Admettons ici que l'on ait déterminé cette fonction. Il est alors très fréquent de vouloir en plus, rajouter des contraintes sur les solutions générées.
Imaginons un problème dans lequel on voudrait discrétiser une variable continue dans un objectif quelconque (arbre de classification ou autre). Il faut trouver des bornes pour les valeurs de la variable qui permettent de définir les différentes classes. Par exemple, si une variable prend ses valeurs entre 0 et 100, on peut construire les classes : [0,20],]20,65] et]65,100]. Il est alors tout à fait probable que l'on voudra imposer un minimum pour l'amplitude et/ou l'effectif des classes. Pour appliquer ces contraintes, on a deux solutions :

- faire en sorte que l'ensemble des opérateurs tienne compte des contraintes et qu'aucune solution violant les contraintes ne puisse être construite,
- laisser les opérateurs de croisement et de mutation construire des solutions ne respectant pas les contraintes mais les pénaliser au niveau de la fitness pour qu'elles aient moins de chances d'être choisies lors de l'application de l'opérateur de sélection.

La première solution a l'avantage d'éviter tout risque d'obtenir une solution ne respectant pas les contraintes. En effet, rien ne permet d'affirmer qu'une telle solution ne pourrait pas avoir une valeur compétitive pour la fonction objectif. Cependant, faire en sorte que les opérateurs de croisement et de mutation n'engendrent pas de solutions violant les contraintes conduit généralement à construire des opérateurs complexes qui finissent par s'éloigner du hasard qu'ils sont censés produire. Nous avons donc, en général, préféré la deuxième solution. Cela implique évidemment de construire des fonctions de pénalisation et de réaliser un équilibre dans la fitness entre la fonction objectif et la (ou les) pénalisation(s). A partir de tout cela, la construction de la fitness dépend très fortement du problème d'optimisation considéré. Nous verrons dans les différentes applications (cf. partie II) comment nous construisons la fitness en fonction du contexte.

1.5.2 Application de l'opérateur de sélection

Une fois que la fitness est définie, on peut la calculer pour l'ensemble des solutions de la population courante. Nous allons maintenant utiliser ces valeurs pour réaliser la sélection. En effet, l'objectif de l'opérateur de sélection est de construire une nouvelle population à partir de la population obtenue après application de la mutation et du croisement. Globalement, on souhaite calculer une probabilité de sélection pour chaque individu de la population qui soit d'autant plus élevée que la valeur de sa fitness est bonne.
Dans ce contexte, la méthode la plus simple pour définir la probabilité de sélection est de la rendre directement proportionnelle à sa fitness. Soit $f(i)$ la valeur de la fitness de l'individu i ($i = 1, 2, \ldots, T_{pop}$). On peut alors définir $p_s(i)$, la probabilité de sélection de l'individu i, comme suit :

$$p_s(i) = \frac{f(i)}{\sum_{k=1}^{T_{pop}} f(k)}.$$

Une fois que les probabilités de sélection sont calculées pour l'ensemble des individus de la population, il reste à s'en servir pour construire la population suivante. La façon la plus

Fig. 1.3 Représentation de l'opérateur de sélection stochastique pur pour des probabilités de sélection proportionnelles à la fitness ($T_{pop} = 5$). Les chiffres indiqués sous les accolades représentent les zones de sélection de chacun des cinq individus.

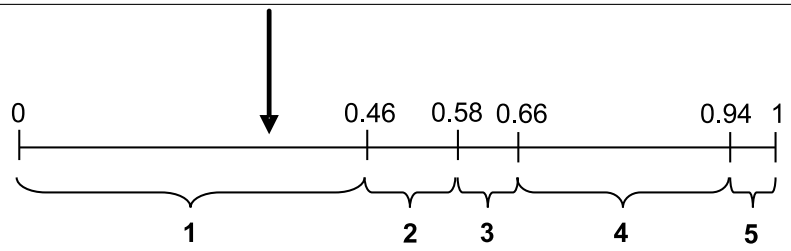
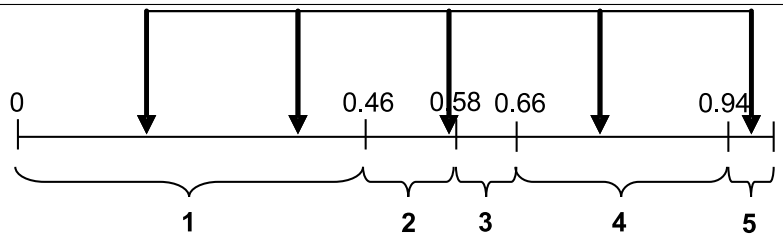


Fig. 1.4 Représentation de l'opérateur de sélection stochastique universelle pour des probabilités de sélection proportionnelles à la fitness ($T_{pop} = 5$). Les chiffres indiqués sous les accolades représentent les zones de sélection de chacun des cinq individus.



simple de faire la sélection est d'appliquer un opérateur de sélection stochastique pur. Pour cela, on construit un segment de longueur 1. Ensuite, on calcule la position de chaque individu de la population sur ce segment en calculant les probabilités de sélection cumulées :

$$position(i) = \sum_{k=1}^{i} p_s(k).$$

Enfin, on génère aléatoirement un nombre dans $[0, 1]$ et on reporte le nombre obtenu sur le segment pour choisir l'individu à sélectionner. Par exemple, si le nombre choisi appartient à l'intervalle $[position(i-1), position(i)]$, on sélectionne l'individu i. Ce procédé est illustré dans la Fig. 1.3. On répète jusqu'à obtenir le nombre d'individus souhaité.

Cependant, par ce processus, le nombre de sélections d'un individu effectivement obtenu peut être assez différent du nombre théorique attendu. Une façon de se rapprocher du nombre théorique attendu est d'utiliser la méthode de *sélection stochastique universelle* (Baker, 1987) qui peut s'apparenter à de l'échantillonnage aléatoire systématique (Lohr, 1999). Cela consiste à sélectionner tous les individus en même temps en utilisant un *pointeur* multiple, dont les branches sont régulièrement espacées, que l'on porte sur le segment précédemment décrit (Fig. 1.4). Hancock (1994) et Hancock (1996) ont montré expérimentalement la supériorité de cette méthode sur le stochastique pur pour se rapprocher du nombre de sélection théorique.

Le fait de baser le calcul des probabilités de sélection directement sur la fitness présente cependant des inconvénients. Tout d'abord, l'opérateur de sélection n'aura pas du tout le même effet sur une fitness prenant ses valeurs dans $[0, 10]$ et sur une fitness prenant ses valeurs dans $[0, 10000]$. Une petite différence de fitness entre deux individus sera beaucoup mieux repérée dans le premier cas. Cet inconvénient peut être surmonté en utilisant la méthode du

scaling proposée par Goldberg (1989b). Cela consiste à fixer deux réels a et b et à transformer la fonction de fitness f en la fonction g de la manière suivante :

$$g = af + b.$$

Pour fixer les paramètres a et b, on impose que les moyennes de f et g soient égales et que la valeur maximum de f soit un multiple de sa moyenne $\overline{f}$:

$$\max(f) = \alpha\overline{f},$$

où $\alpha \in \mathbb{R}$. α est constant au cours des générations mais comme $max(f)$ peut évoluer à chaque itération, il est nécessaire de faire évoluer la fonction g. Comme cela n'est pas très satisfaisant, d'autres méthodes ont été mises au point.
La méthode dite du *tournoi* consiste à extraire m individus de la population courante et de sélectionner le meilleur parmi ces m. Cette étape est répétée autant de fois que d'individus souhaités. On peut cependant relâcher la pression de sélection en ne choisissant la meilleure solution parmi les m qu'avec une probabilité $\pi < 1$.
On peut également utiliser le *rang* à la place de la valeur brute de la fitness (Whitney, 1989). Pour cela, on classe les individus en fonction de leur valeur pour la fitness, la meilleure solution ayant le rang T_{pop}. Alors, pour calculer la probabilité de sélection de l'individu ayant le rang k, on utilise la formule suivante :

$$p_s(k) = \alpha k + \beta,$$

où les réels α et β sont fixés de sorte à ce que :

$$\sum_{k=1}^{T_{pop}} p_s(k) = 1.$$

Une autre condition est nécessaire pour pouvoir déterminer les deux paramètres, cette condition est choisie en fonction de la pression de sélection que l'on souhaite exercer. Par exemple, on peut décider que le meilleur individu doit avoir deux fois plus de chances d'être sélectionné que l'individu de rang médian :

$$p_s(T_{pop}) = 2 \times p_s(T_{pop}/2).$$

Cette méthode a évidemment l'avantage d'être tout à fait indépendante de l'amplitude des valeurs de la fitness. Elle a également l'avantage de s'adapter automatiquement au cours de l'algorithme. En effet, dans les premières générations, si la population initiale est hétérogène, les valeurs de la fitness prendront une grande amplitude mais au fur et à mesure des générations, cette amplitude va diminuer. L'utilisation du rang permettra alors de prendre en compte cette diminution d'amplitude et ainsi, plus l'algorithme avance, plus des petites différences de fitness seront repérées et plus facilement distinguées.

1.5.3 Pression et intensité de sélection

Nous avons parlé précédemment de la pression de sélection que l'on impose en appliquant l'opérateur de sélection. Cette notion est assez instinctive : plus la pression est forte moins les individus les moins adaptés seront sélectionnés. Mais, on peut donner une définition précise de la pression de sélection, Π_s :

Définition 1.1 *La pression de sélection :*

$$\Pi_s = \frac{\mathbf{Pr}(\textit{sélectionner le meilleur individu})}{\mathbf{Pr}(\textit{sélectionner l'individu moyen})}.$$

Ce paramètre est aisément fixé par l'utilisateur, notamment dans le cas de l'utilisation des rangs lorsque l'on doit donner une valeur pour le deuxième paramètre.
Mülenbein & Schlierkamp-Voosen (1994) ont introduit une autre mesure de la sélection appelée *intensité de sélection*. Ils s'appuient pour cela sur une notion de la génétique des populations introduite dans le cadre de l'élevage. Cette mesure est la suivante :

Définition 1.2 *L'intensité de sélection :*

$$I_s(t) = \frac{\overline{f}_s(t) - \overline{f}(t)}{\sigma\left(f(t)\right)},$$

où $\overline{f}_s(t)$ est la fitness moyenne des individus de la génération t, sélectionnés pour construire la population suivante, $\overline{f}(t)$ est la fitness moyenne dans la population et $\sigma\left(f(t)\right)$ est l'écart-type de la fitness dans la population.

L'intensité de sélection permet d'observer l'évolution de la fitness dans la population mais également, grâce à $\sigma\left(f(t)\right)$, elle mesure la diversité de la population courante et donc sa capacité à explorer l'espace des solutions. On peut donc savoir si la population ne perd pas trop vite en hétérogénéité grâce à cet indicateur.

1.5.4 Génération de la nouvelle population

Une fois que l'on a calculé la probabilité de sélection des différents individus et que l'on a décidé comment sélectionner n individus à partir de ces probabilités, il faut encore choisir le nombre de nouveaux individus à inclure dans la nouvelle population. Le cas le plus simple est de partir uniquement de la population de taille T_{pop} obtenue après application des différents opérateurs (qui contient une proportion inconnue d'individus de la précédente génération qui n'ont pas été modifiés) et de sélectionner T_{pop} individus à partir de cette population. Cependant, pour certains, cette méthode semble aberrante car elle ne garantit pas la survie des bonnes caractéristiques apparues. Notons ici que c'est pourtant la stratégie présente dans la nature et dont l'efficacité n'est plus à prouver. Pour se rassurer et accélérer la convergence on peut choisir de garder la meilleure partie de la population précédente et de ne remplacer que les moins bons éléments (De Jong, 1975). Le choix de la proportion à conserver varie entre un individu (c'est ce qu'on appelle l'élitisme) et toute la population moins un individu (Davis, 1991). Enfin, une dernière solution consiste à assembler les individus de la génération précédente et les individus obtenus après application de la mutation et du croisement et à réaliser la sélection de T_{pop} individus dans l'ensemble.

1.5.5 Illustration

Dans l'ACP, on cherche des combinaisons linéaires des variables de départ telles que la variance $V(\mathbf{c}_i)$, de la $i^{\text{ème}}$ composante $\mathbf{c}_i$ soit maximisée et $\mathbf{c}_i$ n'est pas corrélé à $\mathbf{c}_k$ où $k \in \{1, 2, \ldots, i-1\}$. Les axes sont alors les vecteurs propres successifs de $\mathbf{S}$, la matrice de variance empirique des données de départ. Il a été démontré (Mardia *et al.*, 1979) que cette

solution est optimale au sens de plusieurs critères. Dans la fitness de GA-SCA, nous allons donc également chercher à maximiser $V(\mathbf{c}_i)$. Cependant, dans GA-SCA, la non corrélation des composantes successives n'est pas imposée, nous voulons donc également pénaliser la fitness si la corrélation entre composantes n'est pas nulle. Enfin, afin d'obtenir l'interprétation la plus simple possible, on va également chercher à minimiser le nombre de coefficients (entiers) utilisés que l'on rajoutera comme pénalisation du critère. La fitness finale devra donc prendre en compte :

- $V(c_i)$, la variance de la composante courants,
- K_i, le nombre de coefficients de l'axe courant,
- ρ_i, la somme des corrélations entre la composante courante et les précédentes.

La fitness peut s'écrire comme suit :

$$fitness = \alpha_1 \times V(\mathbf{c}_i) + \beta_1 + \alpha_2 \times K_i + \beta_2 + \alpha_3 \times \rho_i + \beta_3,$$

où les coefficients, $(\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3) \in \mathbb{R}^6$, sont déterminés de sorte à ramener chaque terme dans l'intervalle $[0, 1]$ pour que tous les termes aient des amplitudes comparables. Ceci est *simple* car on connaît la valeur optimale pour la première partie du critère (celle obtenue par l'ACP) et pour le nombre de coefficients, il est compris entre 1 et le nombre de variables.

Cependant, comme les composantes ne sont pas nécessairement non corrélées, pour calculer la *vraie* variance expliquée par les composantes successives, nous utilisons le critère de l'équation suivante qui est aussi utilisé dans Rousson & Gasser (2004) :

$$var(k) = tr(\mathbf{SP}_{(k)}(\mathbf{P}_{(k)}^t \mathbf{SP}_{(k)})^{-1} \mathbf{P}_{(k)}^t \mathbf{S}), \text{ with } k = \{1, 2, \ldots, rank(\mathbf{S})\}, \tag{1.1}$$

où $\mathbf{P}_{(k)}$ est la matrice qui contient les coefficients des k premiers axes avec la $i^{\text{ème}}$ colonne composée des coefficients du $i^{\text{ème}}$ axe. $var(k)$ est alors la variabilité expliquée par les k premières composantes.

La construction de l'opérateur de sélection, quant à elle, est totalement indépendante du problème d'optimisation rencontré, il doit juste *savoir* s'il doit maximiser ou minimiser la fitness. Nous avons choisi, pour les raisons présentées dans les paragraphes précédents, d'utiliser la méthode utilisant les rangs des individus de la population. De plus, nous avons ajouté l'étape d'élitisme, c'est-à-dire la sélection systématique du meilleur individu de la population précédente. Il vient remplacer le moins bon individu de la population obtenue après application de l'opérateur de sélection. Nous verrons dans la partie 1 l'impact de l'utilisation de l'élitisme.

1.6 Bilan

Nous avons maintenant en main tous les outils pour construire un AG. Il suffit ensuite de les assembler et de répéter itérativement les étapes de croisement, de mutation et de sélection comme indiqué dans le pseudo-code de la Fig. 1.5.

Pour ce qui est du critère d'arrêt, le plus couramment utilisé est le fait d'atteindre un nombre de générations ou un temps de calcul fixés à l'avance. On peut aussi se baser sur l'évolution de la fitness moyenne dans la population. En effet, une fois que l'AG a atteint un

Fig. 1.5 Pseudo-code de l'AG.

On génère la population initiale.
$e \leftarrow \arg\max_{pop[i],\, i\in\{1,\ldots,T_{pop}} f(i)$
tant que critère d'arrêt non atteint **faire**
    Croisement.
    Mutation.
    Sélection.
    $m \leftarrow \arg\min_{i\in\{1,\ldots,T_{pop}} f(i)$
    $pop[m] \leftarrow e$
    $e \leftarrow \arg\max_{pop[i],\, i\in\{1,\ldots,T_{pop}} f(i)$
fin

optimum, la solution correspondante, par le biais de l'opérateur de sélection, va peu à peu *coloniser* la population et la fitness moyenne de la population (aux mutations près) va peu à peu se rapprocher de la fitness de cette solution optimum. On peut donc construire un critère d'arrêt reposant sur une absence d'évolution de la fitness moyenne dans les populations successives. Mais cela nécessite de fixer une tolérance d'évolution (à cause des mutations) et une fenêtre d'observation (combien de générations on considère). Or, il est très difficile de fixer ce type de paramètres sans avoir fait plusieurs essais. Cela peut donc être utile quand on applique plusieurs fois l'AG sur des données très proches mais les paramètres devront être revus dès que l'on change de problématique.

Illustration
Pour conclure avec l'exemple de GA-SCA qui nous a servi à illustrer le fonctionnement d'un AG, donnons tout de même les résultats obtenus pour deux exemples complémentaires : les données de *pitprop* et des données de reconnaissance d'écriture de chiffres.

Les données *pitprop* ont été introduites dans Jeffers (1967). Elles consistent en treize variables mesurées sur 180 observations. De plus amples détails sont données dans Jeffers (1967). Vines (2000), qui a mis au point une méthode pour réaliser une SCA a montré la difficulté de détecter une structure simple dans ces données. Les résultats obtenus avec l'ACP classique, la SCA de Rousson & Gasser (2004), la méthode de Vines (2000) et GA-SCA sont donnés dans la Tab. 1.1. En haut de la table (matrice des coefficients), les valeurs pour SCA et Vines proviennent des articles originaux (Rousson & Gasser, 2003; Vines, 2000), pour l'ACP, les résultats sont arrondis à deux chiffres après la virgule.
Concernant, les résultats obtenus pour SCA (Rousson & Gasser, 2003), les coefficients sont très faciles à interpréter et ils permettent de mettre en évidence des blocs de variables mais cette méthode impose d'importantes contraintes sur les solutions et par conséquent, les trois derniers axes semblent être trop simples. Notons qu'à part pour le premier axe, il n'existe pas de lien évident entre les coefficients de l'ACP et les blocs de variables obtenus par SCA et GA-SCA (on ne pourrait pas les retrouver en appliquant un seuil aux coefficients de l'ACP). Pour ce qui est de la variabilité, dès la première composante, SCA est déjà assez loin de l'ACP (29.9% pour SCA et 32.5% pour l'ACP). Quant aux corrélations, plus de la moitié des coefficients sont en dehors de l'intervalle $[-0.1, 0.1]$ malgré l'utilisation d'une pénalisation.

Pour ce qui est des résultats de la méthode de Vines, on peut aisément trouver des blocs de variables pour les deux premiers axes mais ensuite, l'interprétation devient de plus en plus difficile. La variabilité est assez proche des résultats de l'ACP mais l'intérêt d'imposer des coefficients entiers est quelque peu perdu. Les coefficients de corrélation sont en dehors de l'intervalle $[-0.1, 0.1]$ pour la moitié des coefficients mais cette méthode n'impose rien sur les corrélations.
Enfin, concernant GA-SCA, les résultats semblent réaliser un bon compromis entre la simplicité et la variabilité. Nous voyons que quatre des six axes ne contiennent que deux coefficients distincts et les deux autres n'en ont que trois. L'interprétation de chaque composante est donc très simple. Enfin, pour les corrélations, seulement deux coefficients (sur quinze) sont en dehors de l'intervalle $[-0.1, 0.1]$.

Le deuxième jeu de données est décrit dans Hastie *et al.* (2001). Il est constitué de 256 variables mesurées sur 7291 individus. Ces variables sont les valeurs en niveaux de gris de chacun des pixels pour une image de dimension 16×16. Quant aux individus, ils sont répartis en groupes avec chaque groupe qui correspond à l'écriture d'un chiffre. Ici, notre objectif n'est pas la discrimination, nous allons donc uniquement nous intéresser aux 658 observations qui représentent le chiffre trois.

D'après la Tab. 1.2, on constate que les coefficients sont distribués plus ou moins symétriquement autour de zéro qui est le mode de la distribution. Ainsi, beaucoup de variables (celles auxquelles le coefficient zéro est appliqué) ne sont pas utilisées pour construire les composantes. Pour ce qui est de la variabilité, les résultats de GA-SCA sont très proches de ceux de l'ACP ce qui s'explique bien par les fortes corrélations existant entre les composantes de l'ACP et celles trouvées par GA-SCA. La Tab. 1.3 montre que les composantes successives issues de GA-SCA sont proches de l'orthogonalité.

Pour compléter la comparaison avec les résultats de l'ACP, la Fig. 1.6 montre la représentation des coefficients des axes pour les deux premières composantes de l'ACP et de GA-SCA en niveaux de gris. Les apparences globales sont très proches mais comme GA-SCA utilise des entiers les résultats sont moins *lisses*.

Enfin, il est intéressant de constater que les blocs de variables obtenus avec GA-SCA n'auraient pas pu être obtenus en réalisant une classification non supervisée des coefficients des axes. Nous avons appliqué aux coefficients du premier axe, l'algorithme K-means (Hastie *et al.*, 2001) avec sept groupes (le nombre total de coefficients obtenus pour le premier axe) et les groupes ne se correspondent pas du tout.

Tab. 1.1 Résultats à 6 composantes (5 pour Vines) pour les données *pitprop* obtenues avec ACP, SCA, la méthode de Vines et GA-SCA. Du haut en bas : les matrices des coefficients, la matrice des correlations entre composantes, le pourcentage cumulé de variabilité permis par chaque composante (corrigées pour la corrélation, voir Eq. 1.1).

ACP						SCA						Vines					GA- SCA					
.40	-.22	.21	-.09	-.08	.12	.45	0	0	0	0	0	1	2	2	-133	620571	1	2	-1	-1	1	0
.41	-.19	.24	-.10	-.11	.16	.45	0	0	0	0	0	1	2	2	-133	620571	1	1	-1	-1	1	0
.12	-.54	-.14	.08	.35	-.28	0	0	.71	0	0	0	0	1	-9	603	-745121	0	2	2	1	-2	0
.17	-.46	-.35	.05	.36	-.05	0	0	.71	0	0	0	0	1	-9	601	-744591	0	1	2	-1	-2	0
.06	.17	-.48	.05	.18	.63	0	.5	0	0	0	.71	1	-2	-5	79	3491021	0	-1	2	1	1	1
.28	.01	-.48	-.06	-.32	.05	0	.5	0	0	0	0	1	0	-11	-333	-2253059	1	-1	2	1	1	0
.40	.19	-.25	-.06	-.22	.00	0	.5	0	0	0	0	1	0	-9	-273	-2268959	1	-1	2	1	1	0
.29	.19	.24	.29	.19	-.06	.45	0	0	0	0	0	1	0	8	250	-1236902	1	-1	-1	1	-2	1
.36	-.02	.21	.10	-.11	.03	.45	0	0	0	0	0	1	1	1	-68	603346	1	1	-1	1	-2	0
.38	.25	.12	-.21	.16	-.17	.45	0	0	0	0	0	1	0	8	224	-240664	1	-1	-1	-1	1	0
.01	.21	-.07	-.80	.34	-.18	0	0	0	0	1	0	0	1	0	20	11701220	0	1	-1	8	1	0
.12	.34	.09	.30	.60	.17	0	0	0	1	0	0	-1	1	-7	-308	666946	0	2	2	-1	1	0
.11	.31	-.33	.30	-.08	-.63	0	.5	0	0	0	-.71	-1	2	3	-79	-3491021	0	2	-1	-1	1	1
1						1						1					1					
0	1					.31	1					.26	1				.001	1				
0	0	1				.17	.11	1				-.07	-.10	1			-.04	.01	1			
0	0	0	1			.22	.18	-.20	1			-.01	.23	-.24	1		.06	.01	.005	1		
0	0	0	0	1		.00	.08	-.13	.03	1		-.13	.004	-.03	.08	1	-.01	-.27	-.02	-.07	1	
0	0	0	0	0	1	-.11	-.02	.09	-.03	.07	1						.35	-.01	-.09	.03	.01	1
32.5	50.7	65.2	73.7	80.7	87.0	29.9	42.0	57.7	65.6	73.9	83.1	30.9	49.2	64.2	71.8	79.6	32.0	49.7	64.3	72.7	79.1	84.7

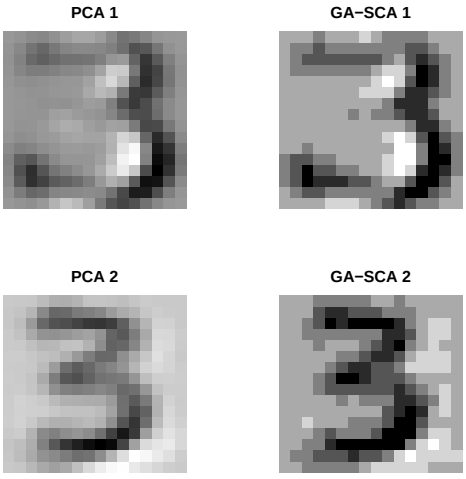
Tab. 1.2 Coefficients obtenus pour les quatre premiers axes des données du chiffre trois. Les colonnes 2 à 10 indiquent le nombre de variables pour lesquelles chaque coefficient est appliqué. *cum var GA-SCA* est le pourcentage de variabilité cumulé permis par chaque composante obtenue avec GA-SCA et *cum var PCA* est l'analogue pour l'ACP usuelle. La dernière colonne indique le coefficient de corrélation entre les composantes de GA-SCA et celles de l'ACP.

loadings	coef									cum var	cum var	correlation
number	-4	-3	-2	-1	0	1	2	3	4	GA-SCA	PCA	GA-SCA/PCA
1	.	.	7	16	104	63	33	20	13	4.43%	4.46%	0.99
2	.	.	2	30	98	53	34	19	20	7.54%	7.56%	0.99
3	10	11	18	38	90	41	13	19	16	10.25%	10.32%	0.95
4	.	19	29	30	60	60	35	23	.	12.63%	12.90%	0.90

Tab. 1.3 Matrice de corrélation pour les quatre premières composantes de GA-SCA pour les données du chiffre trois.

	1	2	3	4
1	1			
2	0.000	1		
3	0.001	0.000	1	
4	0.004	0.000	0.000	1

Fig. 1.6 Images 16×16 correspondant aux deux premiers axes de l'ACP et de GA-SCA pour les données du chiffre trois.



Chapitre 2

Le chemin vers la modélisation et la convergence

Nous venons de décrire les étapes de la construction d'un AG. Nous avons, entre autres, montré combien chaque composante était directement *descendue* de la théorie de l'évolution des espèces. Un tel algorithme a donc une origine très intuitive qui lui permet d'être compris très rapidement. Mais cela ne sous-tend aucun fondement théorique sur lequel on pourrait s'appuyer pour modéliser les AG et éventuellement en déduire des résultats de convergence. Certains écartent rapidement cette question en considérant que les AG sont directement calqués sur la théorie de l'évolution, que ce processus a fait ses preuves et donc, qu'il n'y a aucune raison pour que ce succès ne se reproduise par pour les AG. De fait, c'est le cas ! Mais on peut (et on doit...) montrer de façon plus rigoureuse les bonnes propriétés que l'on observe, ne serait-ce que pour mieux diffuser ces méthodes.
Déjà, Holland (1975) avait réfléchi à cette question : *Pourquoi ça marche ?* Nous allons voir quels éléments de réponse il a essayé d'apporter ainsi que les idées qui ont suivi, aboutissant à une modélisation des AG par une chaîne de Markov. Ceci permet d'appliquer aux AG les nombreux résultats disponibles sur les chaînes de Markov (Ycart, 2002) avec l'inconvénient de conduire à des calculs souvent impossibles à mener à terme, même pour des problèmes très simples, comme nous allons le voir.

2.1 Des difficultés et des Chaînes de Markov

Nous avons évoqué dans le paragraphe 1.1 l'utilisation par Holland des *schémas*. Pour Holland, un schéma est un sous-ensemble de chromosomes qui partagent un ensemble particulier d'allèles définis. Par exemple, si on considère le schéma $\{0, 1, *, *, 1, *\}$, il regroupe les huit chromosomes $\{0, 1, 0, 0, 1, 0\}$, $\{0, 1, 0, 0, 1, 1\}$, $\{0, 1, 0, 1, 1, 0\}$, $\{0, 1, 0, 1, 1, 1\}$, $\{0, 1, 1, 0, 1, 0\}$, $\{0, 1, 1, 0, 1, 1\}$, $\{0, 1, 1, 1, 1, 0\}$ et $\{0, 1, 1, 1, 1, 1\}$. Un schéma définit donc un sous-ensemble de chromosomes, voisins dans un sens défini. On peut les considérer comme des *hyperplans* de l'espace des solutions, ce n'est pas nécessairement un hyperplan au sens usuel, puisqu'on peut laisser libre plus d'une coordonnée. C'est pourquoi on donne aussi le nom de *compétition des hyperplans* à l'étude de l'évolution des schémas au cours des AG. De tels sous-ensembles sont évidemment particulièrement faciles à définir dans le contexte du codage binaire. On peut donner la définition de deux termes liés aux schémas : la longueur d'un schéma est la

distance entre le premier et le dernier locus imposés et l'ordre correspond au nombre de loci imposés. Dans l'exemple ci-dessus, le schéma a une longueur de 4 et un ordre de 3.
A partir de ce concept, Holland a donné dans le *Théorème des schémas* une expression de l'espérance du nombre d'occurrences du schéma S dans la génération $(t+1)$, $\eta(S, t+1)$, en fonction de $\eta(S, t)$, le nombre d'occurrences du schéma S dans la génération précédente. D'une façon tout à fait générale, ce théorème peut s'exprimer de la manière suivante :

Theorème 2.1 Théorème dit des Schémas
Si l'on a $\eta(S, t)$ occurrences d'un schéma S dans la génération t, alors, à la génération $(t+1)$, l'espérance du nombre d'occurrences de ce même schéma est donnée par :

$$\mathbb{E}[\eta(S, t+1)|\eta(S, t)] \geq \{1 - \delta(S, t)\}\sigma(S, t)\eta(S, t),$$

où $\delta(S, t)$ est un terme lié à la destruction d'occurrences de S par application des opérateurs de mutation et de croisement et $\sigma(S, t)$ reflète le rôle de la sélection.

Cette version générique du Théorème des Schémas, bien que peu explicite, a l'avantage de mettre en évidence l'idée fondamentale de Holland qui consistait à considérer les AG comme une compétition entre deux phénomènes : la destruction de solutions par les opérateurs de mutation et de sélection et leur maintien par application de la sélection.
Cependant, pour donner une idée plus précise de ce théorème, nous pouvons expliciter les fonctions $\delta(S, t)$ et $\sigma(S, t)$ dans un cas simple : mutations appliquées locus par locus, opérateur de croisement à un point et sélection proportionnelle à la fitness. Dans ce cas, et pour un schéma S de longueur m et d'ordre k, la probabilité de non-destruction par l'opérateur de croisement est supérieure ou égale à $1 - (\frac{m}{l-1})p_c$ où p_c est la probabilité de croisement et l la longueur des chromosomes. $\frac{m}{l-1}$ représente donc la probabilité pour un croisement de tomber dans la zone définissant le schéma S et $(\frac{m}{l-1})p_c$ la probabilité pour une occurrence du schéma S d'être détruite. En ce qui concerne la mutation, la probabilité de non destruction est supérieure ou égale à $1 - p_m k$ où p_m est le taux de mutation. En effet, la probabilité pour que la mutation ne détruise pas une occurrence de S est $(1-p_m)^k \geq 1-p_m k$. Enfin, la probabilité de sélection d'une occurrence de S vaut, en moyenne, $\frac{f(S,t)}{\overline{f}(t)}$ où $f(S, t)$ est la fitness moyenne des occurrences de S au temps t et $\overline{f}(t)$ est la fitness moyenne de la population au temps t. En combinant ces résultats on obtient le cas particulier du Théorème des Schémas suivant :

Theorème 2.2 Cas particulier du Théorème des Schémas
Si l'on a $\eta(S, t)$ occurrences d'un schéma S dans la génération t, alors, à la génération $(t+1)$, l'espérance du nombre d'occurrences de ce même schéma est donnée par :

$$\mathbb{E}[\eta(S, t+1)|\eta(S, t)] \geq 1 - p_c \frac{m}{l-1} - p_m k \, \frac{f(S, t)}{\overline{f}(t)} \eta(S, t).$$

Des conclusions hasardeuses ont été tirées de ce théorème. Notamment, il est fréquemment avancé que ce théorème permet de montrer que le nombre d'occurrences d'un bon schéma augmente exponentiellement au fur et à mesure des générations. Cependant, il convient de se rappeler que, dans ce théorème, on ne raisonne qu'en terme d'espérance et pour une seule transition. En occultant l'aspect stochastique et la limite temporelle de la dépendance de ce théorème, des erreurs ont été commises alors qu'il apparaît assez évident que, dans une population de taille finie, le nombre d'occurrences d'un schéma ne peut augmenter de façon

exponentielle pendant de nombreuses générations... Dans Holland (1975), il avait même été avancé que les AG étaient une approximation d'une stratégie *optimale*. On a alors cru que les AG étaient la meilleure stratégie de recherche parmi les méthodes connues alors, qui permettaient de réaliser l'optimisation de plusieurs hyperplans en même temps. La notion de schéma a donc été longtemps reprise et approfondie, notamment par Goldberg (1989b) et Michalewicz (1992).
Cependant, cette théorie a, depuis, été souvent critiquée. Notamment dans Vose (1988) qui a critiqué l'approche de Holland et proposé une généralisation. De plus, Wolpert & Macready (1997) ont définitivement mis fin au mythe d'optimalité en démontrant leur *No Free Lunch Theorem* qui, globalement, indique qu'en moyenne, aucune méthode d'optimisation (colonie de fourmis, AG, recuit simulé, tabu search,...) n'est meilleure que la recherche aléatoire et que les succès que l'on obtient proviennent d'une adaptation *manuelle* des méthodes à chaque problème rencontré.
Toutefois, même si l'on est à présent convaincus que les AG ne sont pas une sorte de panacée pour les problèmes d'optimisation, il n'en reste pas moins qu'ils permettent d'obtenir des résultats tout à fait convaincants et qu'il est donc très intéressant de chercher à mieux comprendre leur fonctionnement. Pour cela, il convient d'en donner une modélisation satisfaisante, qui permette de tirer des conclusions, notamment sur leur convergence. Les méthodes de modélisation proposées par Holland, se sont révélées être peu pertinentes et n'ont pas permis d'obtenir des résultats intéressants.

Actuellement, plusieurs pistes sont proposées (voir par exemple, Shapiro *et al.* (1994); Reeves (1994); Peck & Dhawan (1995); Stadler & Wagner (1998) et Reeves (2000)). Nous allons spécialement nous intéresser à la modélisation des AG par le biais des chaînes de Markov. Ceci a été introduit indépendamment par Nix & Vose (1992) et Davis & Principe (1993). L'intérêt du recours à cet outil est le grand nombre de résultats existants que l'on va pouvoir appliquer aux AG. L'inconvénient est, comme nous allons le détailler, la lourdeur des calculs impliqués. Voyons pourquoi et comment utiliser les chaînes de Markov dans le contexte des AG.

L'ensemble des opérateurs que nous avons décrits (mutation, croisement et sélection) comportent des éléments stochastiques, c'est particulièrement vrai pour les opérateurs de mutation et de croisement mais aussi pour la sélection, puisque l'on définit des *probabilités* de sélection. On peut donc classer les AG dans la famille des processus stochastiques. De plus, les états successifs de l'AG (les populations des différentes générations) ne sont pas indépendants les uns des autres. Ce que l'on trouve dans la génération t dépend directement de ce que l'on avait dans les générations précédentes.
Plus précisément, l'état de la génération t ne dépend que de l'état de la génération $(t-1)$. En effet, étant donnée la définition des opérateurs (mutation, croisement et sélection), on constate bien que, quel que soit le passé antérieur de l'algorithme, le contenu de la génération t dépend entièrement (de façon stochastique) du contenu de la génération $(t-1)$. Or, une chaîne de Markov est un processus stochastique qui vérifie la propriété markovienne qui est la suivante : si on note $\{X_t\}_{t\in\mathbb{N}}$ les événements successifs d'un processus stochastique alors,

$$\mathbf{Pr}[X_{t+h} = x_{t+h} | X_s = x_s, s \leq t] = \mathbf{Pr}[X_{t+h} = x_{t+h} | X_t = x_t].$$

On peut donc bien décrire un AG comme une chaîne de Markov.
Ensuite, pour caractériser une chaîne de Markov, il faut connaître son espace d'états, c'est-

Tab. 2.1 Nombre de populations possibles ($\mathcal{N}$) pour un espace des solutions de taille n et une population de taille T_{pop}.

T_{pop}	n	$\mathcal{N}$
10	$2^{10}(\approx 10^3)$	10^{23}
20	$2^{10}(\approx 10^3)$	10^{41}
20	$2^{20}(\approx 10^6)$	10^{102}
50	$2^{50}(\approx 10^{15})$	10^{688}

à-dire l'ensemble des valeurs que peut prendre X_t. Ensuite, il faut connaître l'état initial du processus (qui dépendra de l'initialisation de l'AG) et enfin les probabilités de transition d'un état à l'autre. Nous allons voir que chacun de ces éléments est assez complexe à déterminer dans le cas des AG.

2.1.1 Espace d'états d'un AG.

Avant de connaître l'espace d'états d'un AG, il faut définir ce que l'on considérera comme un état. Dans le cas des AG, on étudie les générations successives, il vient donc naturellement qu'un état est une population donnée : à chaque génération, l'état de l'AG est l'ensemble des solutions courantes, c'est-à-dire la population. L'espace d'états, $\mathcal{X}$, est donc l'ensemble des populations que l'on peut construire (sachant que la redondance à l'intérieur de la population est autorisée). On saisit alors immédiatement qu'un AG sera une chaîne de Markov très complexe, au moins en ce qui concerne l'espace d'états. Pour s'en convaincre, voyons quel est le nombre de populations possibles (soit la taille de l'espace d'états). Soit n la taille de l'espace des solutions (le nombre de chromosomes possibles) et $T_{pop} \in \mathbb{N}$, la taille de la population. Alors, $\text{card}(\mathcal{X}) = \mathcal{N}$, le nombre d'états possibles, est le nombre de combinaisons avec répétition soit :

$$\mathcal{N} = \binom{n + T_{pop} - 1}{T_{pop}},$$

ce qui peut rapidement prendre des valeurs très grandes. Reeves & Rowe (2003) ont donné quelques valeurs qui sont reportées dans la Tab. 2.1. Les chiffres parlent d'eux-mêmes !

2.1.2 Matrice de transition d'un AG.

Pour une chaîne de Markov, la matrice de transition, $\mathbf{\Pi}$, est la matrice qui rassemble les probabilités de passage d'un état à l'autre entre deux temps successifs. On définit, π_{ij}, l'élément de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne de $\mathbf{\Pi}$ comme :

$$\pi_{ij} = \mathbf{Pr}[X_t = i | X_{t-1} = j],$$

où $(i, j) \in \mathcal{X}^2$. Il vient immédiatement que

$$\sum_{k \in \mathcal{X}} \pi_{kj} = 1.$$

Les éléments de $\mathbf{\Pi}$ sont tous positifs ou nuls et les colonnes somment à un, on dit alors que la matrice $\mathbf{\Pi}$ est *stochastique*. De plus, $\mathbf{\Pi}$ sera dite *régulière* s'il existe un $t \in \mathbb{N}$ tel que $\mathbf{\Pi}^t$

ne contient que des réels strictement positifs. Par exemple, si c'est le cas pour $t = 1$, cela signifie que d'une étape à l'autre, tous les états sont *accessibles* quel que soit l'état de départ. Il faut donc que tous les états soient accessibles entre eux, en un nombre de générations fini et indépendant de ces états pour que la matrice soit régulière.
Là encore, non seulement les π_{ij} sont très nombreux ($\mathcal{N} \times \mathcal{N}$) mais en plus, on comprend tout de suite que leur calcul ne sera pas simple. En effet, le passage d'une génération à la suivante doit tenir compte de l'application des mutations, des croisements et de la sélection. Ces opérateurs ainsi que leurs effets doivent donc être modélisés afin de pouvoir déterminer $\mathbf{\Pi}$. Etant donnée la dimension de $\mathbf{\Pi}$, la calculer entièrement n'est pas envisageable dans la grande majorité des cas d'AG. Cependant, la matrice $\mathbf{\Pi}$ possède des propriétés qui permettent de déduire certains résultats.
Posons $\mathbf{p}_t$, le vecteur contenant les probabilités pour chaque état d'apparaître au temps t, le $k^{\text{ème}}$ élément de $\mathbf{p}_t$ est la probabilité pour que l'état k apparaisse. Alors, on a

$$\mathbf{p}_t = \mathbf{\Pi}\mathbf{p}_{t-1}.$$

Ainsi, si l'on part d'une distribution initiale $\mathbf{p}_0$ alors, par récurrence, on obtient,

$$\mathbf{p}_t = \mathbf{\Pi}^t\mathbf{p}_0.$$

Pour savoir ce qu'il se passe au bout d'un temps long (après de nombreuses générations), il faut s'intéresser à $\lim_{t\to\infty} \mathbf{p}_t$. On aura ainsi une idée de la distribution de la population au bout d'un temps long. Une application du théorème de Perron-Frobenius (Reeves & Rowe, 2003) nous apporte des éclairages sur la question :

Theorème 2.3 Application de Perron-Frobenius
Soit $\mathbf{\Pi}$ *une matrice stochastique régulière, alors*

1. $\mathbf{\Pi}^\infty = \lim_{t\to\infty} \mathbf{\Pi}^t$ *existe.*
2. *Les colonnes de* $\mathbf{\Pi}^\infty$ *sont identiques.*
3. *Chaque colonne est identique à l'unique vecteur de probabilité* $\mathbf{q}$ *qui satisfait* $\mathbf{\Pi q} = \mathbf{q}$. $\mathbf{q}$ *est donc un vecteur propre de* $\mathbf{\Pi}$ *associé à la valeur propre 1.*
4. $\mathbf{q} = \lim_{t\to\infty} \mathbf{\Pi}^t\mathbf{p}_0$, *indépendamment du choix de* $\mathbf{p}_0$.
5. *Chaque élément de* $\mathbf{q}$ *est strictement positif.*

$\mathbf{\Pi}$ devant être régulière, ce théorème ne s'applique pas dans le cas où il y a un ou plusieurs états *absorbants*. En effet, un état k est dit absorbant si $\pi_{kk} = 1$ (donc $\pi_{ik} = 0$, $\forall i \neq k$), c'est-à-dire qu'une fois que l'on a atteint l'état k, on ne peut en ressortir. Donc, aucun des autres états n'est accessible à partir de l'un de ces états absorbants. La matrice de transition, dans ce cas, n'est donc pas régulière. Ainsi, le comportement à long terme peut être déduit d'un autre théorème (Reeves & Rowe, 2003) énoncé ci-dessous :

Theorème 2.4 *On suppose que les états d'une chaîne de Markov puissent être réordonnés de sorte que la matrice de transition ait la forme suivante :*

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \tag{2.1}$$

Alors, la matrice $\mathbf{\Pi}$ *est dite réductible et on a :*

$$\lim_{t\to\infty} \mathbf{\Pi}^t = \begin{bmatrix} \mathbf{I} & (\mathbf{I} - \mathbf{S}')^{-1}\mathbf{R}' \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{2.2}$$

Dans cette nouvelle écriture de $\mathbf{\Pi}$, la matrice identité $\mathbf{I}$ est de dimension k, le nombre d'états absorbants. Pour la construire, il suffit donc de placer les états absorbants dans les premières lignes (et donc dans les premières colonnes) et les états non absorbants dans les lignes suivantes. On obtient alors les probabilités pour la population d'être dans tel ou tel état après un très grand nombre de générations. On peut également considérer le cas de sous-ensembles absorbants où la chaîne est piégée dans un sous-ensemble d'états, auquel cas il suffit de remplacer la matrice identité $\mathbf{I}$ par une matrice stochastique de dimension k. Mais dans le contexte de réduction de l'espace d'état qui sera le nôtre dans la suite, la notion d'*état absorbant* sera suffisante.
Il reste cependant à calculer les éléments de $\mathbf{\Pi}$, c'est-à-dire l'ensemble des probabilités de transition d'un état à un autre. Ces éléments sont évidemment extrêmement nombreux et doivent prendre en compte l'influence de tous les opérateurs permettant de passer d'une génération à l'autre (mutation, croisement et sélection). Nous considérerons ce type de calcul dans la partie 3 pour une chaîne de Markov différente. On peut également trouver une écriture de ces probabilités pour des opérateurs spécifiés dans Reeves & Rowe (2003) (p.118-122), où l'espace d'état est identifié au vecteur des entiers représentant le nombre d'occurrences de chaque chromosome possible dans la population courante.
Considérons maintenant deux types de résultats obtenus pour la convergence des AG en utilisant les chaînes de Markov. Nous allons ainsi voir la grande diversité des utilisations possibles de ces outils.

2.2 Divers résultats sur la convergence des AG utilisant des chaînes de Markov.

2.2.1 Utilisation de la théorie de Freidlin-Wentzell

Parmi les premiers résultats réellement fondés sur la convergence des AG, on trouve les travaux de Raphaël Cerf (Cerf, 1994, 1996a,b). Ces résultats se fondent sur la théorie de Freidlin-Wentzell qui étudie les perturbations aléatoires de systèmes dynamiques. R. Cerf a modélisé les AG comme des processus perturbés. En fait, le processus est constitué de l'opérateur de sélection et ce sont les opérateurs de croisement et de mutation qui apportent les perturbations aléatoires. Ces perturbations évoluent au cours du temps en s'amenuisant. Des résultats de convergence vers le ou les optima locaux sont établis sous condition d'avoir une taille de population suffisante. Dans ce contexte, l'opérateur de croisement n'est pas forcément nécessaire.
Dans le cas de populations de très grande taille, R. Cerf s'est aussi intéressé aux chemins pris par la population au cours des générations. Il montre que, quand certains individus atteignent des optima locaux, toute la population a tendance à les suivre, ce qui tend à construire des populations uniformes puis l'application de l'opérateur de mutation permet de faire apparaître de nouveaux optima locaux qui entraînent à nouveau la population. Ce phénomène se repète jusqu'à obtention de l'optimal global.
Ces résultats sont très importants car ils ont permis de modéliser le fonctionnement des AG dans toute leur complexité. Ils sont cependant difficiles à utiliser d'un point de vue pratique. Par exemple, si on s'intéresse à la borne définie pour la taille de la population, on a généralement trop peu d'informations sur la fonction optimisée pour être capables de donner les valeurs de tous les paramètres nécessaires. De même, il faut avoir formulé les opérateurs

de mutation et de croisement dans un cadre théorique complexe. Ce n'est jamais le cas en pratique, où l'opérateur de mutation est défini en fonction du contexte, de façon simple. Pourtant, même si leur utilisation pratique est très difficile, de tels résultats sont indispensables pour comprendre ce qui se passe derrière ce que certains considèrent comme une *boîte noire* et pour donner le cadre théorique nécessaire à une meilleure diffusion de ces méthodes. Notons également que Raphaël Cerf a obtenu d'autres résultats sur les chaînes de Markov qui lui ont notamment permis des rapprochements avec le recuit simulé et qui ont abouti à la construction d'un algorithme génétique qui maintient la diversité dans la population (qui ne converge donc plus vers les populations uniformes) (Cerf, 1996b).

2.2.2 Simplification de l'espace d'état de la chaîne de Markov

Nous avons vu que le problème principal des chaînes de Markov réside dans la complexité des calculs et notamment dans la taille de l'espace d'états. Pour pouvoir tout de même obtenir des résultats sur la matrice de transition, on peut réaliser une agrégation des états (Stewart, 1995). Dans cet objectif, nous allons présenter brièvement le raisonnement de Bhandari *et al.* (1996) qui a permis de démontrer la convergence des AG pour deux conditions nécessaires et suffisantes.
Leur astuce consiste à réaliser une classification des chromosomes possibles en fonction de leur fitness. On obtient alors autant de classes que de valeurs possibles pour la fitness. Ces classes sont ordonnées, toujours en fonction de la fitness. Grâce à cette classification, on va pouvoir réaliser une partition de l'espace d'états de la chaîne de Markov, c'est-à-dire de l'ensemble des populations possibles. Pour cela, on regroupe toutes les populations pour lesquelles le meilleur individu qu'elles contiennent a la même fitness. Cela revient à définir comme fitness de la population entière, la fitness du meilleur individu de la population et à regrouper les populations en fonction de cette fitness. Dans ce cas, la dimension de l'espace d'états se trouve réduite au nombre de valeurs possibles pour la fitness. Notons que, si la fitness prend des valeurs continues, on peut la discrétiser en définissant une précision en dessous de laquelle on considère que les solutions sont équivalentes.
A partir de cette nouvelle définition de l'espace d'états, Bhandari *et al.* (1996) ont construit un raisonnement par récurrence qui leur permet d'assurer qu'un individu de fitness maximum se trouvera dans la population quand $t \to \infty$. Pour cela, la conjonction des deux conditions suivantes est nécessaire et suffisante :

- insertion d'une étape d'élitisme (le meilleur individu de la population courante est systématiquement retenu dans la population suivante),
- l'opérateur de mutation doit être construit de sorte à ce que l'on puisse passer de n'importe quelle solution à toute autre en un nombre donné d'étapes.

On trouvera une modification de la démonstration de cette propriété dans la partie III, où nous étendrons les résultats au cas non homogène (évolution du taux de mutation au cours du temps) et pour un opérateur de mutation dit *dirigé*.

Chapitre 3

Les métaheuristiques : une grande famille

Les algorithmes génétiques entrent dans la catégorie des *métaheuristiques*. Originalement, les métaheuristiques englobaient les méthodes qui combinent des procédures d'optimisation locale et des stratégies à un niveau plus large, permettant de s'échapper des optima locaux. Puis la définition s'est généralisée à toutes les méthodes qui comportent des éléments destinés à sortir des optima locaux, par exemple, en définissant plusieurs structures de voisinage.
Cette famille de méthodes offre une grande diversité. Dans cette partie, les représentants les plus courants de cette famille vont être présentés. Ils seront ensuite mis en perspective les uns par rapport aux autres par l'utilisation d'une sorte de nomenclature générale qui rassemble différentes caractéristiques des métaheuristiques.
Après leur création, et surtout depuis le développement des capacités de calcul des ordinateurs, chacune de ces méthodes a connu des développements, modifications, extensions,... Leur nombre est très important et évolue constamment. C'est pourquoi nous nous limiterons, ici, pour chaque méthode, à l'analyse de la version actuellement considérée comme socle commun à toutes les variantes.
Par définition des heuristiques, ce ne sont pas des méthodes dont on attend *a priori* la convergence garantie vers l'optimum global. Cependant, pour un certain nombre de méthodes (et généralement pour des variantes précises de ces méthodes), il existe des résultats de convergence théoriques qui seront cités le cas échéant.
En outre, l'objectif de cette partie n'est absolument pas de donner un classement des performances des algorithmes décrits. En effet, pour de telles méthodes, il est impossible de définir une performance *absolue*. Elle dépend grandement du problème. Certaines méthodes sont assez simples et ne seront pas adaptées à des problèmes pour lesquels la structure de la fitness dans l'espace des solutions est complexe (beaucoup d'optima locaux par exemple). Mais ces méthodes simples auront l'avantage d'être moins coûteuses en temps de calcul, elles seront donc tout à fait adaptées à la résolution de *problèmes simples*. La nomenclature commune que nous avons choisie permettra en tous cas, de comparer la *philosophie* de chaque algorithme afin d'avoir des éléments de comparaison.
Pour toutes les méthodes, nous présenterons un pseudo-code dans lequel on considérera que l'on cherche à minimiser la fonction f à valeurs dans $\mathbb{R}$.
Les métaheuristiques que nous allons étudier ici sont :

- scatter search,
- tabu search,

- variable neighborhood search,
- guided local search,
- greedy randomized adaptive search procedures,
- ant colony optimization,
- simulated annealing.

3.1 Scatter Search (SS)

L'algorithme de *Scatter Search* (Glover *et al.*, 2003; Glover, 1998) (littéralement, recherche par dispersion) repose sur l'évolution d'une population de solutions (comme les AG). Il procède par combinaison des solutions existantes et retenues. La procédure basique est décrite dans le paragraphe suivant.
On commence par générer un ensemble de b solutions (typiquement, $b \leq 20$), dit *ensemble de référence*. Ces solutions potentielles sont construites de sorte à respecter un certain nombre de critères (valeur de fitness, diversité,...). On crée à partir de cet ensemble une liste de sous-ensembles (qui ne réalise pas forcément une partition de l'ensemble de référence). Par exemple, on peut choisir comme sous-ensembles, toutes les paires possibles de solutions de l'ensemble de référence. On combine alors les éléments de chaque sous-ensemble. Cette combinaison se présente généralement comme une combinaison linéaire des caractéristiques de départ assortie de l'application de contraintes sur les solutions obtenues. Suivant le type de croisement appliqué, on obtient une ou plusieurs solution(s). Si la (ou les) nouvelle(s) solution(s) obtenue(s) est (sont) meilleure(s) (au sens du critère que l'on est en train d'optimiser) que la (ou les) pire(s) solution(s) de l'ensemble de référence, on remplace ces solutions par les nouvelles solutions obtenues.
Les sous-ensembles générés lors d'une étape, sont obligatoirement tous considérés pour le croisement. Une fois que tous les croisements et que les éventuelles mises à jour de l'ensemble de référence ont été effectués, on génère une nouvelle liste de sous-ensembles. On itère ainsi jusqu'à ce qu'aucune des solutions obtenues par croisement des solutions de chacun des sous-ensembles n'ait permis d'améliorer la pire solution de l'ensemble de référence.
On peut alors décrire l'algorithme de SS par le pseudo-code de la Fig. 3.1.

Ceci est une description de l'algorithme de base qui peut être modifié en définissant de façon différente les cinq outils nécessaires à un algorithme de SS :

- une méthode de génération de diversification (pour générer l'ensemble de référence),
- une méthode d'amélioration qui permet éventuellement d'optimiser les solutions obtenues par la méthode précédente et/ou après croisement,
- une méthode de mise à jour de l'ensemble de référence (qui peut ne pas être aussi stricte que celle de l'algorithme de base),
- une méthode de génération des sous-ensembles,
- une méthode de combinaison des solutions.

On peut noter que l'algorithme de SS est souvent associé à un algorithme de *path-relinking*. Il s'agit d'une méthode qui utilise plusieurs solutions de haute qualité (du point de vue du critère à optimiser). Partant de l'une de ces solutions, on crée un chemin, par modifications successives de ses composantes, qui relie cette solution aux autres que l'on a retenues. Un critère de choix doit être défini pour décider, à chaque étape, la composante qui doit être ajoutée. L'objectif d'une telle méthode est d'éviter la *myopie* inévitablement associée à tout algorithme de recherche locale, ici, on sait où l'on va.

Fig. 3.1 Pseudo-code de l'algorithme Scatter Search basique.

Construction de l'ensemble des b solutions de l'ensemble de référence
    $RefSet = \{x^1, x^2, \ldots, x^b\}$.
Evaluer les solutions de *RefSet* et les ordonner de sorte à ce que $x^{(1)}$ soit
    la meilleure au sens de f et $x^{(b)}$ la pire.
iter←VRAI
tant que (*iter*=VRAI) **faire**
    On génère $\mathcal{S}$ l'ensemble des sous-ensembles de *RefSet*
    $S \leftarrow \text{card}(\mathcal{S})$
    iter←FAUX
    pour $i = 1, \ldots, S$ **faire**
        On sélectionne s, le $i^{\text{ème}}$ sous-ensemble de $\mathcal{S}$.
        On applique l'opérateur de combinaison à s, on obtient x.
        si ($x \notin$ *RefSet* **et** $f(x) < f(x^{(b)})$) **faire**
            $x^{(b)} \leftarrow x$ et on réordonne *RefSet*.
            iter←VRAI
        fin
        On supprime s de *Subset*.
    fin
fin

3.2 Tabu Search (TS)

L'algorithme de Tabu Search (recherche avec utilisation de tabous) a été introduit par Glover (1986). C'est une méthode consistant en l'évolution d'une seule solution. Les étapes principales (Gendreau (2003)) sont tout à fait classiques :

- on explore tout le voisinage de la solution courante,
- on choisit la solution de ce voisinage qui minimise le critère à optimiser.

L'originalité de cette méthode est l'introduction d'une mémoire par l'intermédiaire de l'utilisation des tabous. Cela consiste à enregistrer ce qui s'est passé dans les $t \in \mathbb{N}$ étapes précédentes et à interdire que cela se reproduise. A chaque étape, on *décale* donc la mémoire : on enregistre ce qui vient de ce passer et on *oublie* ce qu'il s'est passé il y a $t+1$ étapes. Le but est évidemment de favoriser une large exploration de l'espace des solutions et d'éviter de rester dans un optimum local ou d'y retourner trop rapidement.
On peut alors faire jouer cette mémoire de différentes façons. Il faut tout d'abord choisir ce qu'on enregistre. On peut décider de conserver la mémoire des solutions complètes et interdire de reconstruire une solution que l'on a déjà obtenue. On peut aussi retenir uniquement les mouvements qui ont été effectués et empêcher qu'ils soient inversés. Par exemple, dans le cadre d'une sélection de variables, on peut mémoriser l'événement *ajouter la* $k^{ème}$ *variable*. Cette deuxième solution est évidemment moins coûteuse, notamment pour le temps nécessaire à décider si la nouvelle solution doit être autorisée ou non.
Concernant la longueur de la mémoire, t (c'est-à-dire le nombre d'itérations que l'on prend en compte pour décider ce que l'on interdit), elle est généralement fixée pour toute la durée de l'algorithme. Mais on peut aussi la faire varier au cours de l'algorithme de sorte à favoriser, ou non, une large exploration de l'espace (ce qu'on peut rapprocher des changements

Fig. 3.2 Pseudo-code de l'algorithme basique de Tabu Search .

On génère la solution initiale x_0.
$x \leftarrow x_0$
$f^* \leftarrow f(x_0)$
$x^* \leftarrow x_0$
$\mathcal{T} \leftarrow \emptyset$.
tant que critère de fin non rencontré **faire**
    $x \leftarrow \arg\min_{x' \in (\mathcal{V}(x) \setminus \mathcal{T})}(f(x'))$
    si $f(x) < f^*$ **faire**
        $f^* \leftarrow f(x)$
        $x^* \leftarrow x$
    fin
    On met $\mathcal{T}$ à jour.
fin

de taux de mutation dans l'AG). Enfin, il est possible de générer aléatoirement la taille de la mémoire à chaque étape de l'algorithme.
Le pseudo-code pour l'algorithme TS est présenté dans la Fig. 3.2. On note $\mathcal{T}$ l'ensemble des événements enregistrés (les tabous) associés à la recherche, $\mathcal{V}(x)$ le voisinage de la solution x et $\mathcal{V}(x) \setminus \mathcal{T}$ le voisinage de x privé des éléments tabous.

Le critère de fin est généralement la réalisation d'un nombre prédéfini d'itérations ou bien une absence d'amélioration significative de la fitness pendant un certain nombre d'itérations.

Cette exploration forcée de l'espace évite évidemment de rester bloqué dans un optimum local. Cependant, afin de rendre l'algorithme plus souple, on peut envisager d'autoriser un mouvement tabou si la solution obtenue a une meilleure fitness que toutes les solutions précédemment retenues, c'est ce qu'on appelle un *aspiration criterion*. Cela implique évidemment de calculer la fitness associée à chacune des solutions générées avant de savoir si elles vont être acceptées ou non.
Toutes les étapes précédemment définies sont évidemment coûteuses en temps de calcul. Pour limiter ce temps, on peut éviter de réaliser une exploration exhaustive du voisinage en considérant un échantillon aléatoire de ce voisinage.
Pour améliorer les performances de l'algorithme, le TS est souvent associé à des étapes d'intensification et de diversification. Cela consiste à stopper l'évolution normale de l'algorithme pour changer la façon d'explorer l'espace. Pour cela, citons quelques exemples :

- On peut enregistrer le nombre d'itérations consécutives pour lesquelles les composantes de la solution courante ont été présentes (sans interruption). On peut alors recommencer la recherche à partir de la meilleure solution trouvée en fixant les composantes qui semblent les plus prometteuses.
- On peut aussi changer la structure du voisinage lors de l'étape de recherche locale afin d'autoriser des mouvements plus divers.

Pour ce qui est de la diversification, elle permet de forcer l'algorithme à explorer des régions de l'espace des solutions qu'il a peu visitées jusqu'à cette étape.
Concernant la convergence théorique, il existe des résultats pour des variantes de TS. Ce sont des algorithmes comportant des contraintes pour la structure de la mémoire et du voisinage,

Fig. 3.3 Pseudo-code de l'algorithme VNS.

```
On génère une solution initiale x_0.
x ← x_0
k ← 1
tant que k ≤ k_max faire
        iter←VRAI
        tant que iter=VRAI faire
                On génère x' aléatoirement dans V_k(x).
                On fait une recherche d'optimum local autour de x'.
                On obtient x''.
                si f(x'') < f(x) faire
                        x ← x''
                sinon faire
                        iter←FAUX
                fin
        fin
        k ← k + 1
fin
```

notamment la contrainte de réversibilité des chemins. On peut trouver ces résultats dans Faigle & Kern (1992); Peng *et al.* (1997); Hanafi (2001); Glover & Hanafi (2002).

3.3 Variable Neighbourhood Search (VNS)

La recherche à voisinage variable (Variable Neighbourhood Search) a été introduite assez récemment (Mladenovic, 1995; Mladenovic & Hansen, 1997). Le concept de base est l'exploration systématique d'un ensemble de voisinages différents pendant une étape de recherche locale. Les principes qui ont motivé l'émergence d'une telle méthode sont les suivants :

- un minimum local, dans un voisinage défini, n'est pas nécessairement optimal si on change de voisinage,
- un minimum global est un minimum local par rapport à l'ensemble des voisinages possibles,
- dans beaucoup de problèmes, les optima locaux par rapport à un ou plusieurs voisinage(s) sont assez proches les uns des autres.

Il s'agit donc de construire un ensemble prédéfini de structures de voisinage de taille k_{max}, $\{\mathcal{V}_k\}$, pour $k = 1, \ldots, k_{max}$. VNS combine des étapes déterministes et aléatoires. Le pseudo-code est présenté dans la Fig. 3.3. L'aspect déterministe de VNS réside dans l'étape d'optimisation locale autour de x' alors que la sélection de x' dans $\mathcal{V}_k(x)$ est aléatoire.
L'étape de sélection des solutions est également déterministe et extrêmement rigoureuse. On ne retient x'' que s'il surpasse x en terme de fitness. Or, pour éviter les risques de convergence vers un optimum local et permettre une meilleure exploration de l'espace des solutions, il est préférable d'autoriser l'algorithme à accepter des solutions relativement nouvelles même si la fitness n'est pas réellement améliorée. Pour cela, au lieu de comparer directement $f(x'')$ à $f(x)$, on peut remplacer $f(x'')$ par $f(x'') - \alpha\rho(x, x'')$, où α est un paramètre réel qui gère la tolérance par rapport à $f(x)$ et la fonction ρ mesure la distance entre x et x''. Ainsi,

Fig. 3.4 Pseudo-code pour l'algorithme GLS.

On génère une solution initiale.
pour $i = 1$ à n **faire** $p_i = 0$.
tant que critère de fin non rencontré **faire**
    iter←VRAI
    tant que *iter*=VRAI **faire**
        On recherche autour de x l'optimum local x' de la fonction
            $h(x') = f(x') + \lambda \sum_{i=1}^{n} \mathbb{I}_i\{x'\} p_i$.
        si $h(x') < h(x)$ **faire**
            $x \leftarrow x'$
        sinon faire
            iter←FAUX
        end
    fin
    pour $i = 1$ à n **faire**
        $util_i(x) \leftarrow \mathbb{I}_i\{x\} \frac{c_i}{1+p_i}$
    fin
    $U_m \leftarrow \{i : util_i(x) = \max_{i=1,\ldots,n}(util_i(x))\}$
    pour $(i \in U_m) \quad p_i \leftarrow p_i + 1.$
end

une solution assez éloignée de x aura plus de chances d'être sélectionnée qu'une solution qui ressemble beaucoup à x.

3.4 Guided Local Search (GLS)

Les algorithmes de recherche locale guidée (Guided Local Search) dérivent d'une méthode, issue des réseaux de neurones, appelée GENET (Davenport *et al.*, 1994; Tsang *et al.*, 1999). Son objectif est de diriger l'effort de recherche dans des zones favorables de l'espace des solutions.
Pour cela, il est nécessaire d'associer des caractéristiques à chaque solution. A chaque caractéristique, sont associés un coût et une pénalité. Le coût dépend en général des informations que l'on a *a priori* concernant le problème à résoudre. Quant aux pénalités, elles reflètent les connaissances acquises au cours de l'algorithme.
Etudions le pseudo-code correspondant (Fig. 3.4). Les notations sont les suivantes :

- n : nombre de caractéristiques,
- p_i : pénalité actuellement associée à la caractéristique i,
- λ : paramètre permettant de régler l'influence de la pénalisation,
- $\mathbb{I}_i\{x\}$: fonction indicatrice valant 1 si la caractéristique i est présente dans la solution x et 0 sinon,
- c_i : coût de la caractéristique i.

On peut interpréter $util_i(x)$ comme l'utilité de pénaliser la caractéristique i pour la solution potentielle x. Celle-ci augmente en fonction du coût. En effet, si la présence d'une

Fig. 3.5 Pseudo-code pour l'algorithme GRASP.

$f^* \leftarrow \infty$
pour $i = 1$ à N **faire**
  pour $j = 1$ à M **faire**
    On construit la liste $Cand$ des éléments pouvant être ajoutés à la solution.
    $s \leftarrow \text{card}(Cand)$
    On évalue le coût c_k pour $k = 1, 2, \ldots, s$ associé à l'ajout de $Cand(k)$,
      le $k^{\text{ème}}$ élément de $Cand$.
    $c_{min} \leftarrow \min_{k=1,\ldots,s} c_k$
    $c_{max} \leftarrow \max_{k=1,\ldots,s} c_k$
    $RestCand \leftarrow Cand(\{k : c_k \in [c_{min}, c_{min} + \alpha(c_{max} - c_{min})]\})$
    On choisit un élément m de $RestCand$ aléatoirement.
    On incorpore m dans x.
  end
  On effectue une recherche locale autour de x.
  si $f(x) < f^*$ **faire**
    $x^* \leftarrow x$
  fin
fin

caractéristique est coûteuse, on va faire en sorte d'éviter de l'ajouter. De plus, l'utilité de pénaliser prend en compte la pénalisation actuelle : plus une caractéristique a été pénalisée, moins il est utile de la pénaliser à nouveau. Ainsi, U_m est l'ensemble des caractéristiques pour lesquelles l'utilité de pénaliser est maximale. Ce sont finalement ces caractéristiques que l'on pénalise. Globalement, on cherche donc à explorer des régions intéressantes de l'espace des solutions (associées à des bas coûts) mais la pénalisation permet d'éviter de diriger tous les efforts dans la même direction.

3.5 Greedy Randomized Adaptive Search procedures (GRASP)

GRASP (littéralement, Procédures de recherche adaptative par algorithme glouton randomisé, Feo & Resende (1989, 1995)) est une méthode d'optimisation alternant une phase de construction (par l'algorithme *glouton*) et une phase de recherche locale. La phase de construction permet de construire une solution respectant éventuellement des contraintes imposées par le problème. Le voisinage de cette solution est exploré dans la seconde étape pour trouver un optimum local. La méthode peut être globalement décrite par le pseudo-code présenté dans la Fig. 3.5. On notera N le nombre total d'itérations et M le nombre de d'éléments composant une solution.

On voit ainsi qu'à chaque étape de construction, on se restreint à la liste (appelée RCL pour Restrictive Candidate List) contenant uniquement les meilleurs candidats, c'est-à-dire les éléments dont l'ajout augmente le moins le coût. Cette notion de *meilleurs* candidats est gérée par le paramètre $\alpha \in [0, 1]$. Si $\alpha = 0$, on ne considère que les meilleurs éléments au

sens strict (l'algorithme est dit alors purement glouton) et si $\alpha = 1$, on considère l'ensemble entier des éléments candidats (cas purement aléatoire). Entre ces deux extrêmes, on peut gérer la tolérance pour les éléments à incorporer dans la solution en construction.

3.6 Ant Colony Optimization (ACO)

L'optimisation par colonie de fourmis (ACO) est également une métaheuristique récente (Dorigo & Di Caro, 1999; Dorigo *et al.*, 1999). Cette méthode s'inspire directement du comportement d'une colonie de fourmis qui évoluent dans un milieu et échangent des informations en laissant sur leur parcours une piste de phéromones. Dans ACO, chaque fourmi est un agent qui construit une solution et la piste de phéromones représente une information qui sera utilisée de façon probabiliste pour faire évoluer les fourmis-solutions.
Dans le contexte d'ACO, on considère que les fourmis se déplacent sur un graphe valué. Pour construire ce graphe, on définit un ensemble $\mathcal{E} = \{e^1, e^2, \ldots, e^K\}$ d'éléments pour constituer les solutions (c'est une sorte d'alphabet de base pour la construction des solutions). On construit alors un graphe qui a ces éléments pour sommets. Les fourmis vont alors pouvoir construire les solutions en parcourant les arcs du graphe. A chaque étape, la fourmi va choisir son déplacement de façon probabiliste en se basant sur :

- l'information localement disponible (phéromone laissée par les autres fourmis et coût des éléments),
- sa propre mémoire (qui stocke l'état de la solution en construction),
- les contraintes du problème.

Ainsi, à chaque position, la fourmi a la possibilité d'effectuer autant de mouvements qu'il y a d'éléments connectés à l'élément courant e^j. Alors, pour chacun des déplacements possibles i ($i \in \{1, \ldots, k_j\}$), la fourmi aura une probabilité p_i d'effectuer ce déplacement, probabilité qui dépendra de l'ensemble des caractéristiques précédemment citées. On peut alors essayer de représenter la progression d'une fourmi sous la forme d'un pseudo-code (Fig. 3.6). Les fourmis progressent de façons indépendantes et asynchrones. Une fois que sa solution est complète, chaque fourmi peut effectuer le trajet à rebours et mettre à jour la piste de phéromones en fonction de la qualité de la solution finale obtenue.

Pour éviter une convergence trop rapide (et donc le risque d'obtenir un optimum local), on peut prévoir une *évaporation* de la piste de phéromones au cours du temps. On peut aussi rajouter une sorte d'observateur global qui considère l'ensemble des fourmis et agit en tenant compte de l'ensemble des informations. Par exemple, il peut rajouter des phéromones sur le chemin de la meilleure fourmi.
La convergence théorique d'ACO a été étudiée (Gutjahr (2002)). Ces résultats portent sur une variante d'ACO appelée Graph-based Ant System dans des cas où le facteur d'évaporation ou la borne inférieure de la quantité de phéromones sont dépendants du temps. La démonstration de la convergence passe par une modélisation sous forme de chaîne de Markov.

3.7 Simulated Annealing (SA)

Le Recuit Simulé (Simulated Annealing) est une méthode plus ancienne. (van Laarhoven & Aarts, 1987; Aarts & Korst, 1989) lui consacraient déjà des livres entiers. Mais c'est une

Fig. 3.6 Pseudo-code pour la progression d'une fourmi dans l'algorithme ACO.

```
iter ← VRAI
x ← ∅
tant que iter=VRAI faire
        On construit la liste Cand des mouvements possibles.
        s ← card(Cand)
        pour i = 1, 2, ..., s faire
                On calcule c_i le coût associé à l'ajout de Cand(i) à x,
                   (c_i peut dépendre de l'état actuel de x)
                On enregistre la valeur t_i des phéromones associées à Cand(i).
                On calcule v_i une valeur liée à la violation des contraintes du
                   problème que représente l'ajout de Cand(i) à x.
                On calcule p_i = g(c_i, t_i, v_i) la probabilité de se déplacer vers Cand(i).
        fin
        On choisit le mouvement que la fourmi va effectuer (en fonction des p_i).
        On met à jour la piste de phéromones en fonction du mouvement choisi.
        Si x est complète iter ← FAUX
fin
```

méthode populaire qui continue à faire l'objet de développements (Henderson *et al.*, 2003). Ici, l'analogie est physique. Elle concerne les procédés utilisés en métallurgie qui alternent les phases de recuit et de refroidissement plus ou moins rapides de cristaux pour obtenir des formes plus pures. Dans SA, on imite un processus de refroidissement par l'évolution d'un paramètre de température T qui évolue au cours du temps. La tolérance concernant l'acceptation des solutions générées dépend de ce paramètre comme l'indique le pseudo-code de SA (Fig. 3.7). On note :

- N : le nombre de valeurs différentes prises par T,
- T_i : la $i^{\text{ème}}$ valeur prise par T,
- M_i : le nombre d'itérations pendant lesquelles $T = T_i$,
- $\mathcal{V}(x)$: le voisinage de x.

Pour mettre en œuvre un tel algorithme, il est donc nécessaire de décider d'un programme de refroidissement qui doit être adapté au problème à optimiser. On voit en effet, que T gère au cours de l'algorithme la tolérance pour l'acceptation d'une solution non optimale.
L'algorithme SA est certainement la métaheuristique pour laquelle on trouve le plus de résultats de convergence théorique. Ces résultats reposent sur deux types de modélisation :

- modélisation par une séquence de chaînes de Markov homogènes,
- modélisation par une seule chaîne de Markov non homogène.

Dans le premier cas (Aarts & van Laarhoven (1985); Lundy & Mees (1986); Mitra *et al.* (1986)), chaque chaîne de Markov correspond au temps pendant lequel le paramètre T est constant et on doit supposer que ce temps est suffisamment long pour obtenir le régime stationnaire. Le deuxième cas a été examiné par Gidas (1985); Mitra *et al.* (1986).

Fig. 3.7 Pseudo-code pour SA.

On génère une solution initiale x.
pour $i = 1$ à N **faire**
  pour $j = 1$ à M_i **faire**
    Dans $\mathcal{V}(x)$, on génère x' aléatoirement ou par recherche d'optimum.
    si $f(x') \leq f(x)$ **faire**
      $x \leftarrow x'$,
    sinon faire
      $x \leftarrow x'$ avec probabilité $\exp(\frac{f(x)-f(x')}{T_i})$.
    fin
  fin
fin

3.8 Comparaison des algorithmes

Après cette rapide présentation de différentes métaheuristiques courantes, on s'aperçoit qu'il existe des points communs à certaines méthodes ainsi que des originalités. Afin de pouvoir mettre en perspective ces méthodes, nous avons choisi une série de descripteurs communs qui sont les suivants :

- **Exploration espace** : indique par quels mécanismes l'espace des solutions est exploré et s'il est exploré simultanément par une ou plusieurs solutions en construction,
- **Voisinage exploré** : indique comment est définie la structure des voisinages utilisés ainsi que leur intervention dans l'algorithme,
- **Aléatoire** : indique les étapes de l'algorithme qui incluent des mécanismes aléatoires,
- **Déterministe** : indique les étapes de l'algorithmes qui incluent des mécanismes déterministes,
- **Mémoire** : indique la présence (ou non) d'une mémoire et la méthode utilisée le cas échéant,
- **Coopération** : indique l'utilisation (ou non) d'une coopération entre solutions et la méthode utilisée le cas échéant,
- **Maintien de l'hétérogénéité** : indique les étapes au cours desquelles on cherche à maintenir une certaine hétérogénéité afin de mieux explorer l'espace des solutions en prenant le risque d'engendrer une détérioration des solutions proposées.

La comparaison des métaheuristiques présentées en fonction de ces critères se trouve dans les Tables 3.1 et 3.2. Les points d'interrogation indiquent la présence d'éléments optionnels dans l'algorithme de base. Par exemple, la définition des croisements dans le SS peut faire intervenir le hasard ou non.

Tab. 3.1 Comparaison des métaheuristiques étudiées.

Méthode	Exploration espace	Voisinage	Aléatoire
SS	par plusieurs individus génération population initiale recherche locale	défini pour la recherche locale combinaisons entre individus	génération population initiale génération des sous-ensembles croisement (?)
TS	par un individu génération solution initiale recherche locale	défini pour la recherche locale modifié par les tabous	génération solution initiale
VNS	par un individu génération solution initiale définition des voisinages	plusieurs types de voisinages définis préalablement	génération solution initiale génération d'une solution dans le $k^{\text{ème}}$ voisinage
GLS	par un individu génération solution initiale définition des voisinages	défini pour la recherche locale	génération solution initiale
GRASP	par un individu recherche exhaustive	-	choix dans la RCL
ACO	par plusieurs individus initialisation fourmis (?) déplacements autorisés	déplacements autorisés	initialisation fourmis (?) choix probabiliste du déplacement
SA	par un individu génération solution initiale recherche locale	défini pour la recherche locale	exploration voisinage acceptation probabiliste
AG	par plusieurs individus génération population initiale mutations	défini pour les mutations combinaisons entre individus	génération population initiale mutation croisement sélection probabiliste

Tab. 3.2 Comparaison des métaheuristiques étudiées (suite).

Déterministe	**Mémoire**	**Coopération**	**Hétérogénéité**
recherche locale	hérédité	croisements	-
recherche locale	utilisation des tabous	-	tabous
recherche locale	-	-	-
recherche locale définition pénalisation	pénalisation	-	pénalisation
recherche exhaustive	-	-	choix de la RCL choix probabiliste dans la RCL
-	piste de phéromones	utilisation piste de phéromones	déplacement probabiliste évaporation phéromones
élitisme	-	-	évolution température
élitisme	hérédité	croisement	sélection probabiliste

Deuxième partie

Application à des problèmes d'optimisation complexes : difficulté de la construction de la fitness

Chapitre 1

Les AG appliqués à la discrimination en spectrométrie

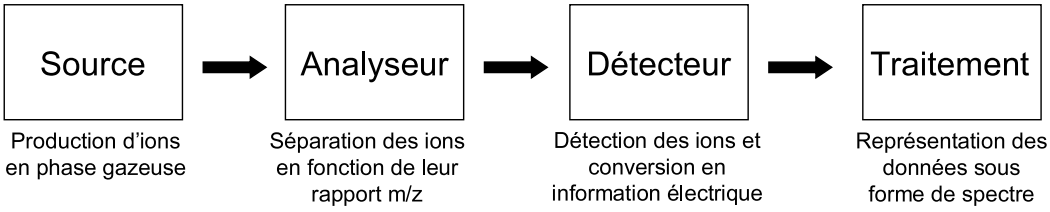
Dans la partie précédente, nous avons décrit, dans le détail, le fonctionnement d'un AG et nous avons montré qu'une bonne maîtrise de la problématique de départ est indispensable à sa mise en place. L'influence du domaine d'application est sensible à toutes les étapes de l'AG mais c'est surtout dans la construction de la fitness qu'il est indispensable de prendre en compte toutes les données du problème. En effet, la traduction de l'énoncé d'une question d'optimisation (dans le domaine biologique dans notre cas) pour réaliser son insertion dans l'AG n'est pas toujours aussi simple que dans le cas de SCA. C'est ce que nous allons étudier dans cette partie.

Actuellement, dans le domaine de la protéomique, deux techniques représentent l'immense majorité des études : la spectrométrie de masse et l'électrophorèse (avec parfois des interactions entre les deux). Nous avons vu que la protéomique a pour but d'explorer le patrimoine en protéines des individus et surtout ses éventuelles modifications lors de changements de conditions, par exemple, lors de l'apparition d'une pathologie. L'objectif final est logiquement de trouver des protéines qui interviennent dans les mécanismes de changement et donc, sur lesquelles il pourrait être intéressant d'intervenir pour empêcher ou favoriser un changement d'état. Dans les deux chapitres suivants, nous allons étudier l'intérêt de l'application des AG à ces deux techniques dans le but ultime de trouver des protéines intéressantes.

1.1 Un problème biologique important : la recherche de biomarqueurs en SELDI-TOF

La spectrométrie de masse est une technique très courante en protéomique. Il existe plusieurs technologies permettant le recours à la spectrométrie. Actuellement, la plus fréquemment rencontrée est le *Surface Enhanced Laser Desorption Ionisation- Time Of Flight* (SELDI-TOF) que l'on peut traduire par *désorption-ionisation laser activée par une surface*. La spectrométrie de masse est, de façon générale, une technique d'analyse qualitative et quantitative. Elle permet l'identification de molécules d'intérêt dans un échantillon, en analysant ses différents constituants après ionisation et parfois fragmentation dans un appareil appelé *spectromètre de masse*. Cette technique utilise les champs électriques et magnétiques afin de séparer les ions obtenus en fonction de leur rapport masse/charge (noté m/z) mesuré en

Fig. 1.1 Schématisation du fonctionnement d'un spectromètre de masse.



daltons (Da). En fait, on devrait plutôt s'exprimer en Da divisés par le nombre de charges fixées (en nombre de protons) mais en spectrométrie SELDI, 99% des protides ne fixent qu'une charge. De plus, l'appareil est toujours étalonné en Da c'est pourquoi on exprime toujours le m/z en Da.
Décrivons brièvement le fonctionnement d'un spectromètre de masse. Cet appareil a été conçu par Joseph John Thomson (prix Nobel 1906 pour ses recherches sur la conduction dans les gaz et le premier modèle de l'atome). Globalement, il comporte une cellule d'ionisation suivie d'un tri (par temps de vol ou par trajectoire) et d'un détecteur. C'est au niveau de ce détecteur que les ions sont comptés, ce qui permet de construire le spectre des masses des ions issus de l'échantillon. L'échantillon peut être introduit sous différentes formes (liquide, solide, en intégralité ou après séparation,...) suivant la technologie utilisée. On peut diviser le spectromètre de masse en trois parties principales :

- la source d'ionisation : c'est la partie de l'appareil où les molécules passent en phase gazeuse et sont ionisées selon une méthode qui dépend de l'application,
- l'analyseur : l'application d'un champ électrique et/ou magnétique permet de séparer les ions obtenus en fonction de leur rapport m/z,
- le détecteur : récupère les ions selon leur rapport m/z et convertit le courant ionique en courant électrique mesurable,
- un ordinateur : même s'il ne fait pas partie intégrante du spectromètre, c'est lui qui traite les données et permet la construction du spectre.

Le fonctionnement d'un spectromètre de masse est schématisé dans la Fig. 1.1.

La principale originalité du SELDI par rapport aux autres technologies de spectrométrie de masse se situe au niveau de l'échantillon introduit. En effet, les échantillons peuvent être déposés sur différentes *surfaces* qui permettent de trier les protéines en fonction de leurs propriétés physico-chimiques (protéines hydrophobes, anioniques ou cationiques, protéines liant des métaux, etc...). Cela permet notamment de pouvoir observer des protéines minoritaires qui sont habituellement masquées par des protéines très abondantes. Pour ce qui est de la source d'ionisation, il s'agit de la désorption-ionisation laser activée par une surface. Quant à l'analyseur, il fonctionne sur la base du temps de vol. La Fig. 1.2 donne un exemple de spectre obtenu par SELDI-TOF. Chaque pic correspond à une détection. On voit immédiatement que selon l'échelle à laquelle on se place, on ne repère pas du tout les mêmes pics. La première étape de l'analyse de spectres SELDI est donc la détection des pics que nous aborderons dans le paragraphe 1.2.3.
La technologie SELDI-TOF est actuellement la technique de spectrométrie la plus utilisée en protéomique essentiellement de par son coût modéré. C'est, de plus, un outil très compact comme le montre la Fig. 1.3.

Fig. 1.2 Exemple de spectre de masse obtenu par la technologie SELDI-TOF : la partie du haut représente un agrandissement de la partie située entre 0 et 15000 Da.

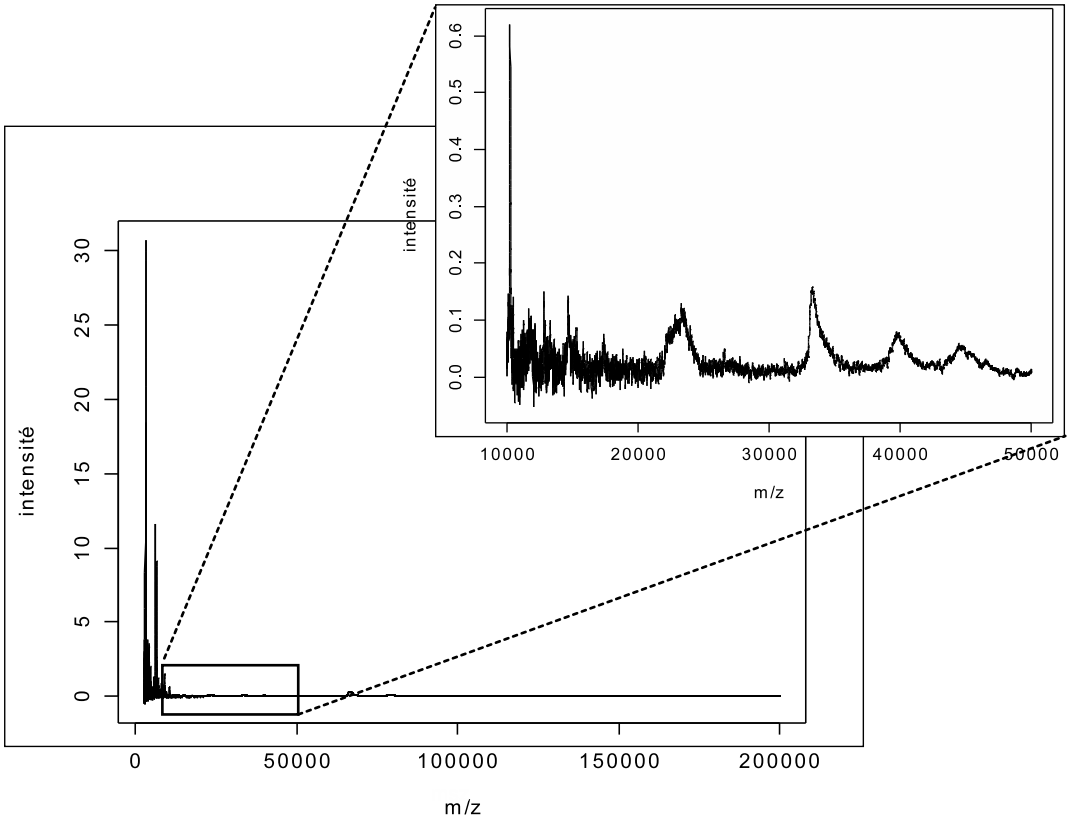


Fig. 1.3 Le spectromètre SELDI-TOF PBS IIc.

L'inconvénient majeur de la technologie SELDI-TOF est son manque de précision concernant la quantification des protéines détectées (l'intensité, située en ordonnée dans la Fig. 1.2), on a mesuré jusqu'à 50% de variabilité (Yasui *et al.*, 2003) ! Or, l'objectif principal de l'application du SELDI-TOF en protéomique est la détection de *biomarqueurs*, c'est-à-dire de molécules caractéristiques d'un état. Pour cela, on peut raisonner en présence/absence, une protéine peut apparaître ou disparaître entre différents états étudiés (c'est-à-dire entre plusieurs groupes de spectres). C'est l'optique abordée dans Yasui *et al.* (2003) mais cela réduit considérablement le champ d'investigation et provoque une perte importante d'information. Il serait beaucoup plus intéressant de pouvoir observer des changements de quantité des protéines entre les différents états mais pour cela, nous devons nous appuyer sur l'intensité mesurée. Dans la majorité des méthodes publiées et utilisées, les biomarqueurs sont recherchés en appliquant des tests de Student ou de Fisher à chaque pic détecté. Cette méthode est évidemment univariée et elle repose entièrement sur une mesure dont on connaît la grande variabilité. Nous avons donc cherché à mettre au point une méthode de discrimination qui tienne compte de tous ces paramètres afin de pouvoir extraire au mieux l'information fournie par les spectres SELDI.
Dans les sections suivantes, nous allons tout d'abord considérer le prétraitement appliqué aux spectres avant toute analyse. Ensuite, nous introduirons la méthode de détection de biomarqueurs que nous avons mise au point et nous l'étendrons à plus de deux groupes de spectres. Une fois ces outils mis au point, nous pourrons les introduire dans un AG. Nous appliquerons ensuite la méthode obtenue à des données réelles. Enfin, nous verrons une autre application des AG dans le contexte de la spectrométrie, la recherche d'intervalles discriminants en spectrométrie proche infrarouge.

1.2 Prétraitement des spectres

1.2.1 Soustraction de la ligne de base

Dans chaque spectre de masse, il existe un niveau de base d'intensité, la *ligne de base* qui dépend de l'échantillon et varie le long du spectre. En fait, cette ligne décrit une sorte de courbe exponentielle pour les petites masses puis devient une fonction linéaire du rapport m/z (Wagner *et al.*, 2003). Nous avons décidé d'utiliser la correction de ligne de base proposée par le logiciel fourni avec les spectromètres SELDI-TOF : le logiciel Ciphergen ProteinChip (Ciphergen, 2002) qui est très largement employé par les utilisateurs de SELDI-TOF.

1.2.2 Normalisation des spectres

La normalisation est une première étape pour corriger le manque de fiabilité sur l'intensité des spectres SELDI. Il existe de nombreuses méthodes de normalisation (Baggerly *et al.*, 2003; Hilario *et al.*, 2003; Prados *et al.*, 2004; Sorace & Zhan, 2003; Tibshirani *et al.*, 2004; Zhu *et al.*, 2003). Cependant, on sait que l'on dépose la même quantité totale de protéines pour chaque spectre, on peut donc considérer que la somme des intensités entre spectres doit être constante, c'est ce que l'on utilise dans la méthode de normalisation dite du *total ion current*, la plus couramment utilisée. Cela consiste à calculer l'intensité moyenne pour l'ensemble des points de chaque spectre, on obtient alors une valeur par spectre et on fait la même chose pour l'ensemble des spectres. Le coefficient de normalisation de chaque spectre est alors obtenu en divisant la moyenne globale par la moyenne individuelle. Ce coefficient

permet, outre la normalisation, de repérer des spectres anormaux. En effet, si le coefficient de normalisation s'éloigne trop de un (en général s'il est inférieur à 0.5 ou supérieur à 1.5), un problème est certainement survenu lors des manipulations et le spectre doit être écarté.

1.2.3 Détection des pics

Certaines méthodes d'analyse de spectres SELDI se basent sur l'ensemble des valeurs du spectre sans extraction des pics. Or, la seule information biologique interprétable est celle donnée par les pics. En effet, chaque pic est censé représenter une protéine ou un protide. Ainsi, même si l'utilisation globale du spectre peut être suffisante pour réaliser la discrimination des spectres, elle ne permet pas de repérer d'éventuels biomarqueurs et donc d'approfondir les résultats obtenus. De plus, si l'on utilise l'ensemble du spectre, pour réaliser la discrimination de nouveaux individus, il sera nécessaire d'obtenir à nouveau le spectre entier alors que si l'on se base sur des pics d'intérêt et que l'on arrive à identifier les substances correspondantes, un dosage de ces substances peut s'avérer suffisant. Nous avons donc développé une méthode d'extraction des pics.
Plusieurs méthodes de détection des pics ont été développées. Le logiciel de Ciphergen, ProteinChip, utilisé par Lee *et al.* (2003), recourt au calcul des intensités et des hauteurs de vallée (c'est-à-dire la différence d'intensité entre le haut et le bas du pic) et sur des seuils dépendant du rapport signal/bruit. Ces seuils sont fixés par l'utilisateur. Yasui *et al.* (2003) et Tibshirani *et al.* (2004) utilisent des méthodes similaires : ils recherchent des maxima locaux dans un voisinage prédéfini et ils calculent l'intensité moyenne dans un voisinage élargi comme définition du bruit local. Les points qui ont une intensité supérieure à ce bruit local et qui sont des maxima locaux sont considérés comme des pics, dont la largeur est fixée à 0.5% du rapport m/z correspondant. Enfin, Coombes *et al.* (2003) identifient tous les maxima et minima locaux en calculant les différences d'intensité entre points successifs des spectres, puis ils définissent le bruit comme la médiane des valeurs absolues de ces différences et, enfin, ils éliminent les maxima locaux dont la différence d'intensité avec le plus proche minimum local est inférieure au bruit défini. Ils rassemblent les maxima locaux séparés de moins de 0.05% du rapport m/z local, les minima locaux des deux côtés du pic sont alors repérés et les différences d'intensité entre ces minima et le maximum correspondant sont calculées. Si les deux valeurs sont inférieures à la moitié du bruit, le pic est éliminé. L'approche que nous avons choisie est légèrement différente. Nous allons montrer de quelle manière et pourquoi.

Description de notre algorithme
Tout d'abord, la première étape est de choisir une définition d'un pic. Nous considérons qu'un pic est un optimum local qui est encadré par deux autres pics qui sont plus ou moins proches de lui. Une fois cette définition posée, nous pouvons construire notre algorithme qui est décrit dans les paragraphes suivants et illustré par la Fig. 1.4.

La première étape consiste à diviser le spectre en plusieurs zones, nous avons choisi de prendre des zones représentant 5% des données mais cette valeur n'a pas d'influence sur les résultats finaux. Dans chaque zone, on identifie le maximum local. Ensuite, on lisse le spectre en utilisant le *super-smoother* décrit dans Friedman (1984). Ce lissage doit être précis car il va permettre de localiser les pics voisins. Il faut considérer ici que les spectres ne sont pas homogènes le long de l'axe des abscisses (Tibshirani *et al.*, 2004). En effet, les données sont plus denses pour les petites masses et la forme des pics changent, ils sont plus étroits et plus hauts pour les petites masses et deviennent de plus en plus bas et larges en allant

Fig. 1.4 Une étape de l'algorithme de détection des pics : les limites des zones dans lesquelles on recherche les pics sont représentées en pointillés fins, les maxima locaux sont repérés par des lignes continues et les limites des pics par des pointillés plus gros.

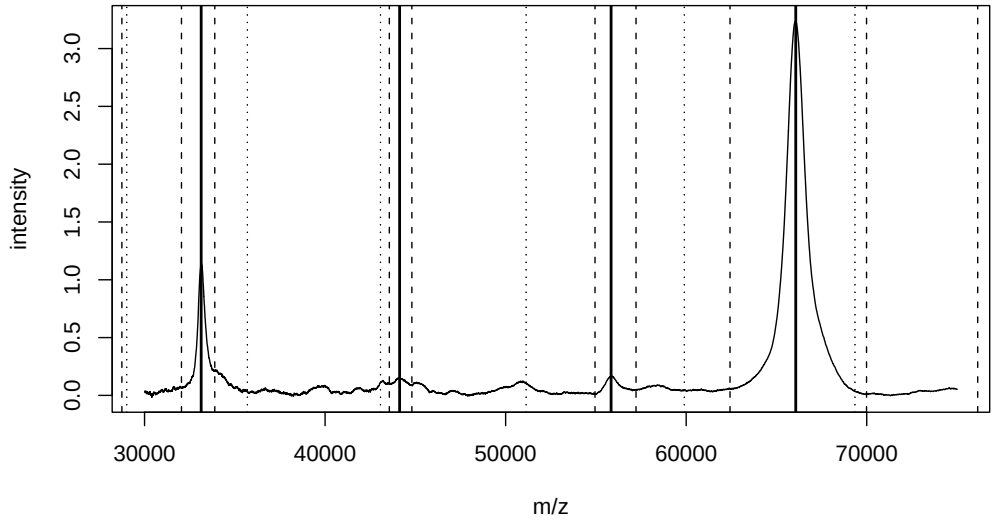


Fig. 1.5 Pseudo-code de l'algorithme de détection des pics.

```
Division du spectre en zones.
tant que le critère d'arrêt n'est pas satisfait faire
    On identifie le maximum local dans chaque zone.
    On lisse le spectre dans chaque zone.
    On repère les changements de signe de part et d'autre du maximum pour fixer les bornes.
    On utilise ces bornes comme limites des nouvelles zones.
fin
```

vers les grandes masses. Le lissage doit donc être appliqué localement. Nous appliquons le super-smoother dans chaque zone et on utilise 0.1% des données pour effectuer le lissage. Ce paramètre n'est pas choisi au hasard. En effet, il s'agit de la précision de la mesure des m/z obtenus avec l'appareil utilisé (Yasui *et al.*, 2003), c'est la seule donnée fiable dont nous disposons. Une fois le lissage réalisé, on calcule la dérivée première du spectre lissé. La dérivée est utilisée pour localiser les bornes du pic : à droite du pic, la borne est fixée à la position où la dérivée redevient positive et à gauche où la dérivée redevient négative. On commence alors une nouvelle itération en utilisant les deux bornes du pic ainsi obtenues pour définir les limites des nouvelles zones dans lesquelles on applique les mêmes étapes (à partir de la recherche du maximum local). Notons que le spectre lissé n'est utilisé que pour fixer les bornes du pic, si l'on utilisait le spectre brut, la moindre irrégularité dans le pic pourrait être considérée comme la borne, mais les maxima locaux, ainsi que toutes les intensités d'intérêt sont mesurés sur le spectre non lissé. On peut décrire l'algorithme de recherche des pics par le pseudo-code de la Fig. 1.5.

L'algorithme est arrêté quand on a atteint un nombre maximum d'itérations. Etant donnée la taille des zones de départ et en accord avec les biologistes, nous avons décidé qu'il n'était pas pertinent d'identifier plus de trente pics par zone, donc on autorise un maxi-

mum de trente itérations. De plus, si une zone est très étroite, on considère qu'il n'est plus possible d'y trouver un vrai pic et on ne l'explore plus. L'algorithme peut aussi s'arrêter quand il n'y a plus de zone assez large.
Enfin, il est possible par cet algorithme d'identifier du bruit comme un pic, en effet, même s'il n'y a pas de pic dans une zone, on pourra toujours identifier un maximum local. Pour surmonter ce problème, on utilise la notion de hauteur de vallée (Prados *et al.*, 2004). Comme indiqué précédemment, il s'agit de la différence d'intensité entre le sommet et le bas du pic. Dans notre cas, le bas du pic est identifié par les bornes du pic, on obtient donc une différence de chaque côté et on calcule la moyenne des deux. Cette valeur est déterminée pour chaque pic. On représente alors la distribution de cette valeur pour tous les pics de tous les spectres. Dans toutes les applications que nous avons réalisées, cette distribution montrait nettement un seuil (voir paragraphe 1.6.1). On élimine tous les pics dont la hauteur de vallée est inférieure à ce seuil.

Il est à noter que notre méthode ne nécessite que peu de paramètres influents : contrairement à d'autres méthodes, il n'y a pas de taille fixée de voisinage pour la recherche des pics et des bornes et la largeur du pic est automatiquement adaptée à la forme de chaque pic. Enfin, la définition du bruit est liée à la hauteur de vallée qui est une mesure classiquement utilisée mais dans notre cas, le seuil utilisé est adapté à chaque jeu de données.

1.2.4 Alignement des pics

Une fois que les pics sont identifiés dans chaque spectre, il est nécessaire de pouvoir déterminer quels sont les pics qui correspondent au même peptide dans différents spectres. En effet, il est impossible de superposer les spectres directement et ce, pour deux raisons : tout d'abord, les spectres ne sont pas mesurés exactement pour les mêmes rapports m/z et ensuite, il existe une incertitude sur la mesure qui est de 0.1% comme nous l'avons précisé dans le paragraphe précédent. Un alignement des pics de l'ensemble des spectres est donc nécessaire.
Pour cela, nous rassemblons les positions de tous les pics issus de tous les spectres en un seul vecteur et nous appliquons une classification hiérarchique ascendante avec lien moyen (Prados *et al.*, 2004; Duda *et al.*, 2001) sur la position des pics. On a alors besoin d'une valeur seuil pour décider à partir de quelle distance on considère que les pics ne représentent pas la même substance. Pour cela, on utilise à nouveau la précision de la mesure permise par l'appareil utilisé : si la différence entre les positions moyennes de deux pics ou groupes de pics est supérieure à deux fois cette précision, on n'autorise pas le regroupement sinon, on considère qu'ils représentent le même peptide. De cette façon, on obtient un seuil qui évolue en fonction de la position sur l'axe des m/z ce qui permet de prendre en compte l'hétérogénéité de cet axe. Pour ne pas risquer de confusion entre les groupes de pics (qui représentent le même peptide) et les groupes de spectres (qui représentent la même condition expérimentale), nous nommerons les groupes de pics des *classes*.
Quand cet alignement est réalisé, on élimine les pics qui apparaissent dans trop peu de spectres : on considère qu'un pic qui apparaît dans un nombre de spectres inférieur à la moitié de l'effectif du plus petit groupe n'est pas susceptible d'avoir un pouvoir discriminant intéressant, il est donc éliminé.
Dorénavant, nous considérerons une matrice $\mathbf{X}$ de taille $(n \times p)$ qui rassemble les intensités des pics dans tous les spectres. Une ligne représente un spectre et une colonne correspond à

une classe de pics. Ainsi, l'élément de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne, x_{ij}, est l'intensité du pic représentant la $j^{\text{ème}}$ classe de pics dans le $i^{\text{ème}}$ spectre. Si une classe de pics n'a pas de représentant dans un spectre, la valeur correspondante est mise à zéro ou au minimum de la zone concernée. On considère ici une expérience concernant n spectres pour lesquels on aurait identifié p classes de pics.

1.3 Construction d'une méthode de discrimination adaptée : la forêt de branches

Dans cette partie, nous allons nous focaliser sur le cas de deux groupes de spectres, par exemple, des spectres correspondant à des individus sains et d'autres à des individus malades. Nous généraliserons la méthode à plus de deux groupes dans le paragraphe 1.4.
Nous avons vu précédemment que l'inconvénient majeur des données de SELDI est leur très grande variabilité en intensité. Pour obtenir une méthode efficace et fiable, il faut nécessairement en tenir compte, contrairement à ce qui est fait dans les nombreuses méthodes qui utilisent les tests de Student et de Fisher ou toute autre méthode qui se base sur les intensités brutes. Mais nous ne voulons pas perdre trop d'information, par exemple en utilisant uniquement le fait qu'un pic soit présent ou pas. Pour cela, nous avons décidé de discrétiser les données mais avec un seuil unique, variable selon les pics, seuil qui sera fixé de sorte à permettre une aussi bonne discrimination que possible, comme dans un arbre de classification (Breiman *et al.*, 1984). Rendre les données binaires est une façon minimale d'utiliser des données peu fiables. On tire plus d'information que si l'on raisonnait simplement sur du présent/absent, on tient compte de l'objectif (ici la discrimination) et on ne *surexploite* pas une information très variable.
De plus, comme il est très peu probable d'arriver à une bonne discrimination avec un seul pic, nous avons choisi de mettre en place une méthode multivariée. Nous allons donc utiliser l'information provenant de plusieurs pics. Pour ce faire et avec l'utilisation d'un seuil, l'arbre de classification semble être tout à fait adapté. Cependant, en SELDI, on a rarement des jeux de données de grande taille. Or, en utilisant un arbre, on prend des décisions (à chaque nœud) à partir de sous-ensembles de spectres de plus en plus petits, qui ne sont rapidement plus représentatifs de l'échantillon global et on risque de surajuster rapidement l'échantillon d'apprentissage. Si l'on applique ensuite l'arbre trouvé à un nouveau jeu de données, le résultat risque d'être très peu pertinent. Nous avons donc décidé d'utiliser *en parallèle* des arbres de décision à un seul niveau (Qu *et al.*, 2002). Un tel arbre partage l'échantillon d'apprentissage en deux sous-échantillons en utilisant un seuil sur une variable (ici l'intensité d'une classe de pics). Dans Qu *et al.* (2002), les différents arbres sont construits l'un après l'autre de sorte à ce que les faiblesses des précédents soient compensées par les suivants. Dans notre cas, nous allons les choisir simultanément, cela peut sembler moins performant mais nous allons optimiser, conjointement, les pics utilisés et les seuils par AG et tenir compte de la parcimonie (donc du nombre de pics utilisés). Par cette méthode, on favorisera donc les petits ensembles de pics qui seront complémentaires. Tout cela conduit à une méthode multivariée où chaque décision est prise sur l'ensemble de l'échantillon d'apprentissage, ce qui est tout à fait favorable quand on ne dispose que de petits échantillons.
Pour appliquer à la recherche de biomarqueurs en SELDI-TOF, chaque arbre à un niveau correspond au *vote* d'un pic comme nous allons le décrire dans les paragraphes suivants.

L'objectif est alors de construire ce que nous appellerons un *comité*, c'est-à-dire un ensemble de pics avec leur seuil de discrétisation associé. Nous autoriserons à utiliser entre 1 et N_m pics (donc arbres à un niveau) pour constituer un comité. Attendu que l'on cherche à obtenir un modèle parcimonieux, on voudra avoir peu de pics et donc, fixer un nombre maximum de pics dans les comités n'est pas très contraignant et nous verrons plus loin que cela est très utile dans l'AG. Dans un premier temps, nous allons considérer que le comité est connu et nous allons décrire précisément la méthode de discrimination.

Modélisation

Pour déterminer les règles de décision du comité, nous n'allons utiliser qu'une partie des n spectres de l'échantillon, qui contiennent n_1 spectres du groupe 1 et n_2 spectres du groupe 2. Soit $\mathcal{L}$ l'ensemble de ces spectres. On considère un comité C, contenant N_C pics et leurs seuils. Pour chaque pic m_i^C $(i = 1, 2, \ldots, N_C)$ du comité, il faut déterminer à quel groupe un spectre l $(l \in \mathcal{L})$ sera affecté suivant l'intensité du pic m_i^C dans ce spectre, $x(l, m_i^C)$, par rapport au seuil associé, $z_{m_i^C}$. Le vote du pic m_i^C dans le cas où $x(l, m_i^C) > z_{m_i^C}$ sera noté $V^+_{m_i^C}$ $(V^+_{m_i^C} \in \{1, 2\})$ et $V^-_{m_i^C}$ si $x(l, m_i^C) \leq z_{m_i^C}$ $(V^-_{m_i^C} \in \{1, 2\})$. Ces votes sont calculés comme suit :

Définition 1.1 *Calcul du vote des pics :*
Cas des spectres situés au-dessus du seuil :

$$V^+_{m_i^C} = \arg \max_{g=\{1,2\}} \left(\frac{1}{n_g} \sum_{l \in L \cap g} \mathbb{I}\{x(l, m_i^C) > z_{m_i^C}\} \right),$$

Cas des spectres situés au-dessous du seuil :

$$V^-_{m_i^C} = \arg \max_{g=\{1,2\}} \left(\frac{1}{n_g} \sum_{l \in L \cap g} \mathbb{I}\{x(l, m_i^C) \leq z_{m_i^C}\} \right),$$

On peut alors calculer ces votes pour les N_C pics. Soit $\mathcal{T}$ l'ensemble des spectres non utilisés. Une fois les règles de décisions déterminées, on peut calculer le taux de bien classés en les appliquant aux spectres de $\mathcal{T}$. Pour cela, on calcule l'affectation $A(l)$ de chaque spectre $l \in \mathcal{T}$ $(A(l) \in \{1, 2\})$.

Définition 1.2 *Calcul de l'affectation d'un spectre à un groupe :*

$$A(l) = \arg \max_{g=\{1,2\}} \left(\sum_{i=1}^{N_C} \mathbb{I}\{v_{m_i^C}(l) = g\} \right),$$

où

$$v_{m_i^C}(l) = \mathbb{I}\{x(l, m_i^C) > z_{m_i^C}\} V^+_{m_i^C} + \mathbb{I}\{x(l, m_i^C) \leq z_{m_i^C}\} V^-_{m_i^C}.$$

Si, pour un spectre, on a autant de votes pour un groupe que pour l'autre, on considère qu'il est mal classé.
Etudions maintenant la généralisation à plus de deux groupes.

1.4 Généralisation à plus de deux groupes : utilisation du *pairwise coupling*

Quand on a plus de deux groupes, on peut toujours utiliser des arbres de classification mais nous en avons vu les inconvénients dans la partie précédente. On pourrait aussi utiliser des méthodes telles que l'analyse discriminante mais cela reviendrait à travailler sur les intensités brutes, ce qui n'est pas souhaitable non plus. Nous voulons donc conserver notre approche par seuils d'intensité. Pour $k = 3$ groupes, on pourrait envisager de fixer deux seuils pour chaque pic et on obtiendrait trois décisions d'affectation pour chaque pic, ainsi, un pic serait capable de discriminer seul les trois groupes. Mais si $k > 3$, cela devient plus difficile. C'est pourquoi nous avons choisi d'utiliser la méthode dite du *pairwise coupling* (Wu *et al.*, 2004) à notre problématique. Cela consiste à réaliser toutes les discriminations deux à deux des groupes, à en déduire des probabilités d'affectation aux classes et à estimer des probabilités d'affectation globales en combinant les probabilités issues des comparaisons deux à deux.

1.4.1 Calcul des probabilités d'affectation par couple de groupes

Afin de réaliser les comparaisons deux à deux, il faut disposer d'un comité par couple de groupes. Il existe $k(k-1)/2$ couples de groupes distincts pour k groupes, il faut donc choisir $k(k-1)/2$ comités. L'objectif de cette étape est de calculer, pour chaque spectre, $l \in \mathcal{T}$:

$$r_{ij}(l) = \mathbf{Pr}(l \in i | l \in \{i,j\}, \mathbf{X}, C^{ij}),\ i = 1, 2, \ldots, k,\ i < j \leq k, \tag{1.1}$$

où $\mathbf{X}$ est la matrice contenant les intensités des pics, C^{ij} est le comité choisi pour comparer les groupes i et j, il est constitué de $N_{C^{ij}}$ pics et seuils. On obtient ainsi une matrice $\hat{\mathbf{R}}(l)$ de dimension $(k \times k)$ où $\hat{r}_{ij}(l)$, l'élément de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne, correspondant à l'estimation de $r_{ij}(l)$ est calculé comme suit.

Définition 1.3 *Calcul des estimateurs des probabilités d'affectation des spectres pour les comparaisons deux à deux :*

$$\hat{r}_{ij}(l) = \frac{1}{N_{C^{ij}}} \sum_{n=1}^{N_{C^{ij}}} \mathbb{I}\{v_{m_n}^{ij}(l) = i\} \qquad si \quad i < j, \tag{1.2}$$

$$= \frac{1}{N_{C^{ij}}} \sum_{n=1}^{N_{C^{ij}}} \mathbb{I}\{v_{m_n}^{ij}(l) = j\} \qquad si \quad i > j, \tag{1.3}$$

$$= 1 \qquad si \quad i = j, \tag{1.4}$$

où $v_{m_n}^{ij}(l)$ est le vote décidé par le $n^{ème}$ pic du comité choisi pour discriminer les groupes i et j (l'indice C^{ij} a été omis dans $m_n^{C^{ij}}$ pour alléger l'écriture).

1.4.2 Calcul des probabilités d'affectation globales

Une fois que les probabilités d'affectation ont été calculées pour chaque couple de groupes, il existe plusieurs méthodes (Wu *et al.*, 2004) pour calculer les probabilités d'affectation globales pour le spectre l,

$$\mathbf{p}(l) = \{p_1(l), p_2(l), \ldots, p_k(l)\}.$$

Pour alléger l'écriture nous allons remplacer $\mathbf{p}(l)$ par $\mathbf{p}$ dans la suite, en sachant qu'un vecteur $\mathbf{p}$ est calculé pour chaque spectre $l \in \mathcal{T}$. De même, nous remplacerons $\hat{r}_{ij}(l)$ par $\hat{r}_{ij}$.
La méthode choisie ici a été introduite par Wu *et al.* (2004) qui ont réalisé plusieurs simulations et ont comparé les performances de différents estimateurs des probabilités. Cette méthode semble être réellement robuste, notamment pour des changements de structures de probabilité. Cela consiste à calculer le vecteur $\mathbf{p}$ tel qu'il minimise :

$$\min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j\neq i} (\hat{r}_{ji}p_i - \hat{r}_{ij}p_j)^2, \quad \text{avec } \sum_{i=1}^{k} p_i = 1 \text{ et } p_i \geq 0,\ \forall i \in \{1,\ldots,k\}. \tag{1.5}$$

Le théorème (1.1) montre que les contraintes de positivité des p_i sont redondantes.

Theorème 1.1 *Le problème (1.5) est équivalent à*

$$\min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j\neq i} (\hat{r}_{ji}p_i - \hat{r}_{ij}p_j)^2, \quad \textit{avec } \sum_{i=1}^{k} p_i = 1. \tag{1.6}$$

Démonstration
Il suffit de montrer que toute solution optimale $\mathbf{p}$ de (1.6) satisfait $p_i \geq 0,\ \forall i \in \{1,\ldots,k\}$. Si cela n'est pas vrai, on suppose, sans perte de généralité :

$$p_1 \leq 0, \ldots, p_r \leq 0, p_{r+1} > 0, \ldots, p_k > 0,$$

où $1 \leq r < k$ et il y a un $1 \leq i \leq r$ tel que $p_i < 0$. On définit alors une autre solution possible de (1.5) :

$$p'_1 = 0, \ldots, p'_r = 0, p'_{r+1} = p_{r+1}/\alpha, \ldots, p'_k = p_k/\alpha,$$

où $\alpha = 1 - \sum_{i=1}^{r} p_i > 1$.
Avec $r_{ij} > 0$ et $r_{ji} > 0$, on obtient

$$(r_{ji}p_i - r_{ij}p_j)^2 = (r_{ji}p'_i - r_{ij}p'_j)^2 \geq 0, \text{ si } 1 \leq i, j \leq r,$$

$$(r_{ji}p_i - r_{ij}p_j)^2 = (r_{ij}p_j)^2 > \frac{(r_{ij}p_j)^2}{\alpha^2} = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ si } 1 \leq i \leq r,\ r+1 \leq j \leq k,$$

$$(r_{ji}p_i - r_{ij}p_j)^2 \geq \frac{(r_{ji}p_i - r_{ij}p_j)^2}{\alpha^2} = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ si } r+1 \leq i, j \leq k.$$

On en déduite que

$$\sum_{i=1}^{k} \sum_{j:j\neq i} (\hat{r}_{ji}p_i - \hat{r}_{ij}p_j)^2 > \sum_{i=1}^{k} \sum_{j:j\neq i} (\hat{r}_{ji}p'_i - \hat{r}_{ij}p'_j)^2.$$

Cela est en contradiction avec le fait que $\mathbf{p}$ est une solution optimale de (1.6) puisque la solution obtenue avec les p'_i est meilleure. Donc une solution contenant au moins un élément strictement négatif ne peut être une solution optimale de (1.6). Donc la contrainte, $p_i \geq 0,\ \forall i \in \{1,\ldots,k\}$ est inutile. □

Le problème (1.6) peut ensuite être écrit sous la forme :

$$\arg\min_{\mathbf{p}} 2\mathbf{p}'\mathbf{Q}\mathbf{p}, \tag{1.7}$$

où $\mathbf{Q}$ est la matrice symétrique de dimension $(k \times k)$ définie par :

$$Q_{ij} = \sum_{l:l\neq i} \hat{r}_{li}^2, \qquad \text{si } i = j, \tag{1.8}$$

$$= -\hat{r}_{ji}\hat{r}_{ij} \qquad \text{si } i \neq j, \tag{1.9}$$

Alors, $\forall \mathbf{v} \neq \mathbf{0}_k$, on a $\mathbf{v}'\mathbf{Q}\mathbf{v} = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}(r_{ji}v_i - r_{ij}v_j)^2 \geq 0$, donc $\mathbf{Q}$ est semi-définie positive. Par conséquent, $\mathbf{p}$ est un minimum global si et seulement si il satisfait la condition d'optimalité suivante :

Il existe un scalaire b tel que :

$$\begin{bmatrix} \mathbf{Q} & \mathbf{1}_k \\ \mathbf{1}_k' & \mathbf{0}_k \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{0}_k \\ 1 \end{bmatrix} \tag{1.10}$$

où $\mathbf{Q}\mathbf{p}$ est la dérivée de (1.7) et b le multiplicateur de Lagrange associé à la contrainte $\sum_{i=1}^{k} p_i = 1$. Alors, la solution de (1.5) peut être obtenue en résolvant le système linéaire (1.10).

1.5 Insertion dans l'AG

Dans la partie précédente, on considérait les comités comme connus, nous allons maintenant considérer l'utilisation des AG pour optimiser ces comités. Pour cela, détaillons la construction des différentes étapes de l'AG qui ont été définies dans la partie 1.6.

1.5.1 Codage des solutions

Nous venons de voir que l'objectif est maintenant de rechercher un comité constitué de couples de pics et de seuils associés qui permette d'obtenir une bonne discrimination des pics. Nous avons donc transformé le problème biologique (recherche de biomarqueurs en SELDI-TOF) en un problème de sélection de variables et de discrétisation. Pour insérer ce problème dans l'AG, il faut tout d'abord définir et coder ce que représente une solution à ce problème d'optimisation. Dans cette application, une solution représente un comité. Pour la décrire, il est nécessaire d'avoir l'identifiant associé à chaque pic du comité ainsi que la valeur du seuil qui lui est associé.
Pour cela, deux types d'alphabets sont nécessaires, un pour les numéros des pics, $\mathcal{A}_p$ et un pour les valeurs des seuils de chacun des N_C pics du comité, $\mathcal{A}_{t_i}$ avec $i = 1, 2, \ldots, p$. Concernant les numéros des pics, si l'on dispose de p pics (nombre de colonnes de la matrice $\mathbf{X}$) alors $\mathcal{A}_p = \{1, 2, \ldots, p\}$. Pour les valeurs des seuils, notons $\tilde{\mathbf{x}}_j$ le vecteur ordonné des valeurs d'intensité prises par le pic j dans l'ensemble des spectres. Alors, les valeurs de seuils, pour le pic j, seront nécessairement comprises entre $\min(\tilde{\mathbf{x}}_j)$ et $\max(\tilde{\mathbf{x}}_j)$. De plus, il n'est pas nécessaire de définir plus d'une valeur de seuil entre deux valeurs successives de $\tilde{\mathbf{x}}_j$ et quelle que soit cette valeur, l'impact sur la classification sera le même puisque l'on aura toujours autant de spectres de part et d'autre du seuil. On choisit donc de prendre comme valeurs de

Fig. 1.6 Exemple de codage d'un comité contenant trois pics : la première valeur représente un identifiant de pic et la deuxième, le seuil associé puis on trouve deux autres identifiants chacun suivi de la valeur de son seuil. Enfin, on ajoute le nombre de NA nécessaires pour obtenir une longueur de $2 \times N_m$.

110	11.47	188	6.21	180	1.53	NA	NA	...

seuils possibles l'ensemble des valeurs moyennes entre deux valeurs successives de $\tilde{\mathbf{x}}_j$. Enfin, on rajoute la plus petite valeur de la série. En effet, dans le cas où on a plusieurs spectres pour lesquels le pic n'est pas présent, la plus petite valeur sera zéro et on pourra alors avoir zéro comme seuil, ce qui permettra de faire la différence entre présent et absent. On a donc, au maximum, n valeurs de seuils par pic (on rappelle que n est le nombre de spectres), c'est le cas si toutes les valeurs d'intensité observées dans les différents spectres sont différentes. On obtient $\mathcal{A}_{t_i} = \{t_{i1}, t_{i2}, \ldots, t_{ic_i}\}$ où $c_i \leq n$.
De plus, pour que les opérateurs soient plus simples à appliquer, il est tout à fait souhaitable que toutes les solutions soient représentées par des vecteurs de la même longueur. Pour cela, on peut imposer un nombre maximum de pics, N_m, dans le comité. Ainsi, si on a moins de N_m pics dans le comité, il suffira de compléter le vecteur avec des NA (qui signifie *not available* dans le logiciel R © et représente généralement les données manquantes) par exemple. Nous avons évoqué précédemment qu'un nombre de pics restreint était plus souhaitable : si on arrive à une bonne classification avec peu de pics, c'est que ces pics sont tout à fait représentatifs de la population alors qu'une très bonne classification obtenue avec de nombreux pics est susceptible de relever du surajustement. Donc, fixer un nombre maximal de pics n'est pas gênant. Il est donc nécessaire de rajouter NA dans les alphabets qui deviennent alors :

Définition 1.4 *Alphabets nécessaires à la construction de l'AG pour la recherche de biomarqueurs en SELDI-TOF :*

$$\mathcal{A}_p = \{1, 2, \ldots, p, NA\}$$

$$\mathcal{A}_{t_1} = \{t_{11}, t_{12}, \ldots, t_{1c_1}, NA\} \text{ où } c_1 \leq n$$

$$\ldots$$

$$\mathcal{A}_{t_i} = \{t_{i1}, t_{i2}, \ldots, t_{ic_i}, NA\} \text{ où } c_i \leq n$$

$$\ldots$$

$$\mathcal{A}_{t_p} = \{t_{p1}, t_{p2}, \ldots, t_{pc_p}, NA\} \text{ où } c_p \leq n$$

Finalement, une solution (donc un comité) sera représentée par un vecteur de longueur $2 \times N_m$: N_m valeurs pour les identifiants des pics et N_m valeurs pour les seuils correspondants. Notons que si, pour une solution, on n'a que N_C pics dans le comité, il y aura $2 \times N_C$ valeurs réelles dans le vecteur et $2(N_m - N_C)$ NA. Un exemple est présenté dans la Fig. 1.6. On obtient donc un problème combinatoire ayant un nombre de solutions fini (puisque tous les alphabets sont finis) mais très grand (voir chapitre 1.2 de la partie 3).

1.5.2 Initialisation

Dans le cas du SELDI, nous n'avons aucune information *a priori* concernant le nombre de pics à inclure dans le comité et encore moins sur les meilleurs pics à considérer. Nous allons donc générer la population initiale de sorte à ce qu'elle soit le plus hétérogène possible. Les paramètres à générer sont :

- N_C, le nombre de pics du comité : $N_C \in \{1, 2, \ldots, N_m\}$,
- $\mathbf{u}$, le vecteur des identifiants des pics sélectionnés : $\mathbf{u} \in (\mathcal{A}_p)^{N_C}$,
- $\mathbf{v}$, le vecteur des seuils correspondants aux pics sélectionnés : $v_i \in \mathcal{A}_{t_i}$,

où v_i est le $i^{\text{ème}}$ élément de $\mathbf{v}$ ($i = 1, 2, \ldots, N_C$) et $\mathcal{A}_{t_i}$ est l'ensemble des valeurs de seuils possibles pour le pic d'identifiant u_i, le $i^{\text{ème}}$ élément de $\mathbf{u}$.
Le nombre de pics est généré aléatoirement dans $\{1, 2, \ldots, N_m\}$ avec équiprobabilité. Pour le choix des pics, les pics ayant une plus large amplitude de valeurs d'intensité ont plus de chances de présenter un intérêt pour la discrimination. Nous avons donc choisi de sélectionner, pour la population initiale, les pics avec une probabilité proportionnelle à leur amplitude : plus l'amplitude des intensités d'un pic est importante plus la probabilité de le sélectionner dans la population initiale est grande et inversement. Cette solution a également l'avantage de défavoriser les très petits pics qui pourraient être liés à du bruit non éliminé lors de l'extraction des pics.
Concernant le choix du seuil, afin de ne pas pénaliser les comités qui comportent peu de pics, nous avons décidé d'initialiser la population avec un seuil considéré comme optimal pour chaque pic pris individuellement. Pour cela, considérons deux groupes de spectres d'effectifs respectifs, n_1 et n_2. Alors, la proportion $\pi_{mg}(j)$ de spectres dans le groupe g pour lesquels le pic d'identifiant m a une intensité supérieure à z_m^j, le $j^{\text{ème}}$ seuil possible pour ce pic est déterminée de la manière suivante :

$$\pi_{mg}(j) = \frac{1}{n_g} \sum_{i \in g} \mathbb{I}\{x(i,m) > z_m^j\}, \tag{1.11}$$

où $x(i,m)$ est l'intensité du pic m dans le spectre i et $i \in g$ signifie que l'on somme pour tous les spectres appartenant au groupe g.
Le seuil optimal, $\hat{z}_m^{\hat{j}}$ est choisi parmi les c_m valeurs possibles pour le pic m de sorte que :

$$\hat{j} = \arg \max_{j=1,\ldots,c_m} |\pi_{m1}(j) - \pi_{m2}(j)|. \tag{1.12}$$

Ce seuil optimal est automatiquement appliqué quand un pic est utilisé pour construire la population initiale ou dans les générations suivantes, quand un nouveau pic est ajouté. Cela a pour but d'éviter une importante chute systématique de la valeur d'une solution quand on ajoute un pic. Le seuil optimal dans un comité donné n'est évidemment pas nécessairement le seuil optimal déterminé quand le pic est seul, il pourra évoluer au cours des générations. L'initialisation de la population peut alors être décrite par le pseudo-code de la Fig. 1.7.

1.5.3 L'opérateur de mutation

Nous avons vu dans le paragraphe 1.3 que l'opérateur de mutation doit fournir le hasard nécessaire à une bonne exploration de l'espace des solutions. Il doit assurer que tous les points de cet espace peuvent être atteints et ainsi éviter une convergence rapide vers un optimum local. Comme présenté au paragraphe 1.3, nous allons utiliser un taux de mutation, $p_m(t)$

Fig. 1.7 Pseudo-code de l'initialisation de l'AG pour la recherche de biomarqueurs en SELDI-TOF.

On calcule les probabilités de sélection de chaque pic en fonction de son amplitude.
pour $m = 1, 2, \ldots, p$ **faire**
    On détermine $\hat{z}_m^{\hat{j}}$ le seuil optimal du pic m.
pour $i = 1, 2, \ldots, T_{pop}$ **faire**
    $N_c \leftarrow rand(\{1, 2, \ldots, N_m\})$,
    pour $j = 1, 2, \ldots, N_c$ **faire**
        On génère l'identifiant u_j en fonction des probabilités de sélection.
        On lui associe son seuil optimal.
    fin
On construit la solution générée et on l'insère dans la population.
fin

qui varie au cours des générations : il est élevé au début de l'algorithme pour permettre une large exploration puis il diminue pour autoriser une convergence et enfin il augmente à nouveau pour laisser l'algorithme *sortir* d'un éventuel optimum local.
Pour s'assurer que tout l'espace des solutions pourra être atteint par les solutions, il faut permettre tous les changements possibles, dans ce cas, ce sont les suivants :

- suppression d'un pic : un pic est aléatoirement choisi parmi les pics présents dans le comité et supprimé de la solution (avec son seuil),
- ajout d'un pic : un nouveau pic est aléatoirement choisi et son seuil optimal lui est associé,
- déplacement d'un seuil : pour un des pics du comité choisi au hasard le seuil présent est remplacé aléatoirement par un autre seuil possible pour ce pic.

Nous rappelons que l'utilisation du seuil optimal a pour but d'éviter une diminution systématique trop importante de la valeur d'une solution en cas d'ajout d'un pic.
Pour appliquer l'opérateur de mutation, tout d'abord, on choisit au hasard les individus de la population qui vont subir la mutation. Ce choix tient bien entendu compte du taux de mutation courant. Notons ici que le taux de mutation s'applique aux individus de la population et non à leurs éléments pris un par un. Cela permet de tenir compte de la solution d'un point de vue global, essentiellement ici pour favoriser la parcimonie des solutions en prenant en considération le nombre de pics déjà présents dans la solution courante. Dans une solution, tous les éléments des vecteurs de solution n'ont pas la même signification (identifiant ou seuil), on peut donc choisir le type de mutation à appliquer. Pour cela, on définit la probabilité, p_s, d'éliminer un pic de la solution considérée, la probabilité, p_a, d'ajouter un pic et la probabilité, p_d, de déplacer un pic. Pour une solution contenant np pics, ces probabilités sont définies comme suit (Green, 1995) :

$$p_s(np) = \lambda \times \min(1, \frac{pois(np-1)}{pois(np)}), \tag{1.13}$$

où $pois(k)$ est la probabilité pour le nombre de pics d'être égal à k par une distribution de Poisson de paramètre $\frac{N_m}{2}$, où N_m est toujours le nombre maximum de pics dans le comité (DiMatteo *et al.*, 2001). Ce paramètre de la loi de Poisson est choisi de sorte à favoriser la parcimonie du modèle, c'est-à-dire un petit nombre de pics (toujours pour éviter le surajus-

Fig. 1.8 Pseudo-code de l'application de l'opérateur de mutation pour l'AG de recherche de biomarqueurs en SELDI-TOF.

```
pour i = 1, 2, ..., T_pop faire
    On génère μ ∈ [0, 1].
    si μ ≤ p_m faire
        np ← nombre de pics dans la solution i.
        On calcule les probabilités p_s(np), p_a(np) et p_d(np).
        On choisit le type de mutation en fonction de ces probabilités.
        On applique la mutation.
        On remplace la solution i par la solution obtenue.
    fin
fin
```

tement). Le coefficient λ est utilisé pour assurer que $p_r \geq 0$ (voir Eq. 1.15). Ce paramètre doit alors être fixé en fonction de N_m, par exemple, pour $N_m = 10$, une valeur de $\lambda = 0.4$ était appropriée.
De la même façon,

$$p_a(np) = \lambda \times \min(1, \frac{pois(np+1)}{pois(np)}), \tag{1.14}$$

et

$$p_d(np) = 1 - (p_s + p_a). \tag{1.15}$$

Grâce à ces probabilités, un type de mutation est choisi et appliqué à la solution considérée comme décrit précédemment. La nouvelle solution remplace alors l'ancienne dans la population. Le déroulement global de l'application de l'opérateur de mutation est décrit dans le pseudo-code de la Fig. 1.8. Rappelons qu'après application des mutations, on n'a aucune idée *a priori* sur leurs conséquences en termes de fitness.

Si on considère les conditions nécessaires et suffisantes de convergence définies par Bhandari *et al.* (1996) on ne retrouve pas exactement la première condition énoncée. En effet, par cette définition des mutations par individu qui permettent de prendre en compte la vision globale de la solution pour *orienter* les mutations, on ne peut pas arriver de n'importe quelle solution à n'importe quelle autre en n'importe quel nombre donné d'étapes. La partie 1 permettra de généraliser les résultats de convergence de Bhandari *et al.* (1996) à notre cas de mutations dirigées et on verra de plus, dans la partie 3, comment ce type de mutation peut permettre une modélisation intéressante de l'AG.

1.5.4 L'opérateur de croisement

Dans cette application, la complémentarité entre les pics d'un comité (donc d'un individu de la population) est primordiale, l'échange de pics entre solutions est donc à même de faire apparaître de nouvelles interactions entre différents pics. Ces dernières peuvent être positives pour la fitness ou au contraire détruire une symbiose déjà efficace. Voyons comment le croisement, qui remplit ce rôle, est appliqué dans cet exemple.
La longueur des solutions est assez faible, elle est de $2 \times N_m$ et un paramètre N_m grand n'est

pas du tout souhaitable. Nous avons donc choisi d'appliquer un opérateur de croisement à deux points pour lequel on intervertit, entre les parents, le fragment central (Jeffries, 2004). Pour choisir les parents, on réalise une permutation aléatoire des individus de la population comme décrit et pour chaque couple d'individus pris successivement, on décide de les croiser ou non en fonction du taux de croisement, p_c. Considérons un couple que l'on a choisi de croiser et soient np_1 et np_2 leurs nombres de pics respectifs. On choisit avec équiprobabilité Np_c, le nombre de pics à intervertir,

$$Np_c \in \{1, 2, \ldots, \max(np_1, np_2)\}.$$

Il ne reste plus qu'à choisir le point de début du croisement qui doit être choisi dans l'ensemble

$$\{1, 2, \ldots, \min\left[(N_m - Np_c + 1), \max(np_1, np_2)\right]\}.$$

Ce point correspond au premier point échangé, si on prend comme début la première position, cela revient à ne faire un croisement qu'à un point de rupture.
Enfin, les pics et leurs seuils associés sont échangés entre les deux parents conduisant à deux enfants qui les remplacent dans la population. Ce processus est appliqué itérativement à tous les couples choisis pour subir le croisement. On obtient effectivement un opérateur qui ne tient pas compte de l'amélioration ou de la détérioration des enfants par rapport aux parents. Voyons maintenant la définition de la fitness et son utilisation pour la sélection qui, seule, permet de répondre au problème d'optimisation.

1.5.5 Définition de la fitness et opérateur de sélection

Rappelons ici que l'objectif de cet AG est de trouver des biomarqueurs parmi les pics présents dans les spectres. Il faut donc repérer des comités de pics permettant une discrimination efficace des groupes de spectres. La fitness doit donc, en priorité, tenir compte du taux de bien classés permis par chaque comité selon la méthode décrite dans les paragraphes 1.3 et 1.4. Par ailleurs, nous avons déjà indiqué qu'un petit comité de bonne qualité était plus à même de comporter des pics réellement discriminants pour l'ensemble de la population qu'un comité constitué de nombreux pics, qui surajuste l'échantillon d'apprentissage et limite sa possibilité de généralisation. Nous voulons donc également que la fitness prenne en compte la parcimonie du modèle obtenu sous forme de pénalisation. Attendu que l'on a déjà introduit un nombre maximal de pics, que l'on a défavorisé les comités d'effectif important lors de l'application de l'opérateur de mutation et que l'on ne connaît pas le nombre de pics optimal, la pénalité introduite, $\rho(np)$, est simplement une fonction linéaire du nombre de pics np :

$$\rho(np) = a \times np + b \text{ avec } (a, b) \in \mathbb{R}^2. \tag{1.16}$$

Dans ce cas, nous allons chercher à maximiser la fitness. Les réels a et b sont donc calculés de sorte à ce que $\rho(np)$ soit minimum pour un grand nombre de pics et maximum pour peu de pics. De plus, afin de faciliter l'équilibrage des deux termes de la fitness, chacun sera ramené dans l'intervalle $[0, 1]$. Finalement, a et b sont déterminés tels que

$$\rho(1) = 1$$
$$\rho(N_m) = 0$$

Quant au taux de bien classés, τ, il est calculé pour chaque individu de la population comme indiqué précédemment. Il est par définition dans $[0, 1]$ avec la valeur un représentant le cas le plus favorable où tous les individus de l'échantillon test local sont bien classés. Si cela est nécessaire, on rajoute un paramètre $c \in \mathbb{R}$ pour réaliser l'équilibre entre les deux termes et on obtient une fitness de la forme :

$$\text{fitness} = \tau + c \times \rho(np) \tag{1.17}$$

Après application des opérateurs de croisement et de mutation, la population est constituée d'individus n'ayant subi aucune transformation et d'individus ayant subi croisement et/ou mutation. Chaque opérateur étant appliqué de manière probabiliste, les nombres d'individus non transformés et transformés sont compris tous deux entre 0 et T_{pop}. On peut alors calculer la valeur de fitness pour chacun de ces individus pour réaliser la sélection. Nous avons choisi d'appliquer la méthode de sélection qui repose sur les rangs des individus pour les raisons indiquées au paragraphe 1.5. On a donc :

$$\mathbf{Pr}[\text{sélectionner l'individu de rang } k] = \alpha \times k + \beta,$$

avec α et β choisis tels que :

$$\sum_{k=1}^{T_{pop}} \alpha \times k + \beta = 1$$

et

$$\mathbf{Pr}[\text{sélectionner la meilleure solution}] = 2 \times \mathbf{Pr}[\text{sélectionner la solution de rang médian}].$$

De plus, pour satisfaire à la deuxième condition nécessaire et suffisante de convergence définie par Bhandari *et al.* (1996), on applique l'étape d'élitisme qui consiste à automatiquement sélectionner le meilleur individu de la population, de sorte à ce que la fitness du meilleur individu de la population ne puisse pas diminuer au cours des générations. Rappelons que le *pire* individu de la population obtenue après application de l'opérateur de sélection est remplacé par l'individu issu de l'élitisme (la taille de population est ainsi maintenue constante). Finalement, l'AG obtenu est décrit dans le pseudo-code de la Fig. 1.9. On voit que la séparation des spectres en échantillon d'apprentissage et échantillon test pour la discrimination est réalisée à chaque étape. En effet, si cette séparation restait la même durant tout l'AG, on n'arriverait qu'à surajuster l'échantillon test obtenu alors que, dans notre cas, si une solution réalise le surajustement de l'échantillon test obtenu au cours d'une génération, comme cet échantillon change à la génération suivante, elle obtiendra une mauvaise fitness et sera défavorisée lors de la sélection. Ainsi, les solutions d'une même génération (qui sont comparées par l'intermédiaire de l'opérateur de sélection) sont appliquées au même échantillon test mais cet échantillon change d'une génération à l'autre. Cette méthode permet de favoriser les comités dont les performances sont bonnes sur la majorité des échantillons tests et donc qui ont de bonnes chances d'être généralisables. Pour ce qui est du critère, on considère dans un premier temps qu'il s'agit d'un nombre maximal de générations atteint, ce nombre est fixé empiriquement.

Fig. 1.9 Pseudo-code de l'AG pour la recherche de biomarqueurs en SELDI-TOF, $pop[i]$ est le $i^{\text{ème}}$ individu de la population courante.

```
On génère la population initiale.
e ← arg max_{pop[i], i∈{1,...,T_pop}} f(i)
tant que critère d'arrêt non atteint faire
        Croisement.
        Mutation.
        Division de l'échantillon en échantillon d'apprentissage et échantillon test
              pour la discrimination.
        Calcul de la fitness pour chaque individu de la population.
        Sélection.
        m ← arg min_{i∈{1,...,T_pop}} f(i)
        pop[m] ← e
        e ← arg max_{pop[i], i∈{1,...,T_pop}} f(i)
fin
```

1.5.6 Généralisation à plus de deux groupes

Globalement, le déroulement de l'AG pour $k > 2$ groupes est le même que celui décrit dans le paragraphe précédent. Cependant, comme nous utilisons le *pair-wise coupling*, $k(k-1)/2$ populations doivent être construites, chacune d'elles permettant la discrimination d'un couple de groupes de spectres.
Pour ce qui est de l'initialisation, les seuils optimaux individuels sont déterminés pour chaque comparaison et chaque population est construite comme indiqué au paragraphe 1.5.2. Il est nécessaire que toutes les populations aient la même taille, T_{pop}. Les étapes de croisement et de mutation sont appliquées indépendamment sur chacune des $k(k-1)/2$ populations comme décrit dans les paragraphes 1.5.4 et 1.5.3. En revanche, l'étape de sélection prend en compte simultanément les $k(k-1)/2$ populations. En effet, ce sont les probabilités d'affectation aux groupes globales qui nous intéressent.
Tout d'abord, pour chaque solution i de chaque population ($i = 1, 2, \ldots, T_{pop}$), on calcule la probabilité d'affectation pour le couple associé comme indiqué dans le paragraphe 1.4.1. On obtient ainsi les probabilités $\hat{r}_{ij}(l)$ pour $i = (1, 2, \ldots, (k-1))$ et $j = (1, 2, \ldots, k)$ calculées sur l'échantillon test associé à la génération courante. On peut alors calculer les probabilités d'affectation globales, $\mathbf{p}(l)$ (cf. paragraphe 1.4.2). Le spectre l est alors affecté au groupe pour lequel la probabilité globale est maximale, on répète pour tous les spectres de l'échantillon test et on obtient le pourcentage de bien classés associé à cette solution (qui est donc constituée de $k(k-1)/2$ *sous-solutions*) en comparant avec les vrais groupes qui sont connus.
Pour ce qui est de la parcimonie, on prend en compte la somme des nombres de pics utilisés dans les $k(k-1)/2$ comités, les paramètres du terme de pénalisation doivent être recalculés de sorte à ce que $\rho(k(k-1)/2) = 1$ et $\rho(N_m \times k(k-1)/2) = 0$. Tous les individus de la ligne i de chacune des populations auront donc la même valeur de fitness. Finalement, l'opérateur de sélection est appliqué exactement comme décrit dans le paragraphe 1.5.5.

Tab. 1.1 Paramètres utilisés pour appliquer l'AG de rechercher de biomarqueurs.

Taille de la population : $T_{pop} = 200$
Nombre de générations : $N_{gene} = 200$
Nombre maximum de pics dans le comité : $Nmax = 10$
Taux de mutation initial : $p_m = 0.8$
Taux de croisement : $p_c = 0.7$

1.6 Une première approche de la convergence : l'observation

Nous allons maintenant appliquer la méthode décrite précédemment à deux jeux de données. Tout d'abord, nous étudierons un jeu de données public à deux groupes concernant des femmes atteintes ou non de cancer des ovaires. Ce jeu de données, malgré quelques inconvénients, a été traité par de nombreuses techniques, ce qui nous permettra de mesurer l'efficacité de notre méthode par comparaison. Le deuxième jeu de données concerne trois groupes de patients atteints de différents troubles neurologiques, il nous permettra d'appliquer l'extension à plus de deux groupes de notre méthode. Ces deux jeux de données nous permettront également de voir comment l'étude de la population finale de l'AG permet d'avoir un certain nombre d'informations sur la convergence de l'algorithme. La Tab. 1.1 donne les valeurs des paramètres utilisés pour les deux applications.

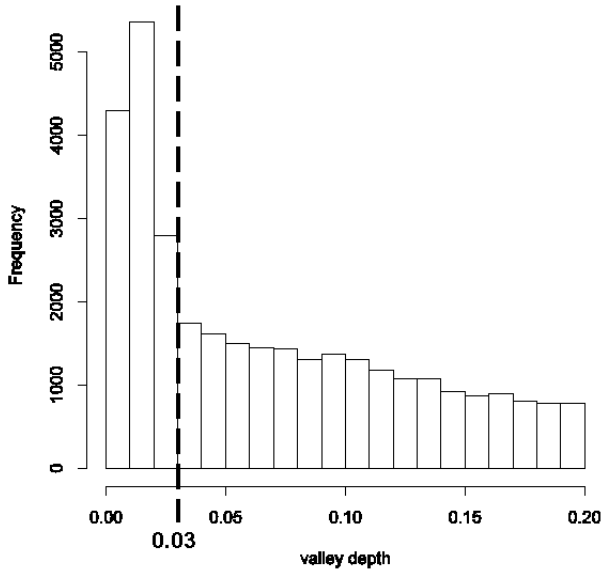
1.6.1 Données publiques de cancer de l'ovaire

Nous avons utilisé les données à *basse résolution* issues d'un instrument Ciphergen qui est identifié sous le nom de 8-7-02 data sur le site de la NCI-FDA (http ://home.ccr. cancer.gov/ncifdaproteomics/ppatterns.asp). Ces données sont constituées de 162 échantillons concernant des patientes atteintes de cancer des ovaires et de 91 échantillons de contrôle. Chaque spectre contient 15154 points avec un rapport m/z variant entre 0 et 20000 Da. De ces 253 échantillons, 46 spectres contrôles et 81 spectres de cancer sont choisis au hasard pour constituer l'échantillon d'apprentissage et les spectres restant (45 contrôles et 81 cancers) seront utilisés pour tester la qualité du modèle obtenu.
Ce jeu de données a déjà été analysé par différentes méthodes telles que les k-plus proches voisins (Zhu *et al.*, 2003), statistiques non paramétriques et analyse discriminante pas à pas (Sorace & Zhan, 2003), AG correspondant à l'algorithme décrit par Petricoin *et al.* (2002) (Jeffries, 2004), SVM (Jong *et al.*, 2004) et analyse logique (Alexe *et al.*, 2004). De plus, Liu *et al.* (2002) ont comparé plusieurs méthodes pour la sélection de variables et la discrimination de ce jeu de données.

1.6.1.1 Prétraitement

Comme nous l'avons décrit dans le paragraphe 1.2, la première étape consiste à extraire et aligner les pics des différents spectres. Ceci est fait pour l'ensemble des spectres des échantillons d'apprentissage et de test. Après application de l'algorithme de détection des pics, environ 350 pics potentiels ont été identifiés par spectre. Les hauteurs de vallée ont alors été calculées et leur distribution est représentée dans la Fig. 1.10. On identifie aisément une

Fig. 1.10 Distribution des hauteurs de vallées des pics issus de l'algorithme de détection pour les données publiques de cancer des ovaires.

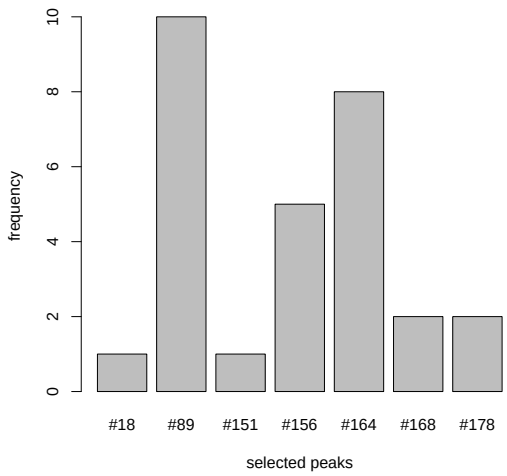


rupture dans la distribution vers 0.03, c'est cette valeur que l'on choisit donc comme seuil pour la définition du bruit. Par application de ce seuil, 12500 pics sont éliminés (environ 50 pics par spectre).
L'étape suivante est l'alignement des pics identifiés entre spectres. La classification hiérarchique ascendante avec lien moyen est appliquée et conduit à identifier 1097 classes de pics. Comme nous ne retenons que les classes contenant plus de 46 spectres (la moitié de l'effectif du plus petit groupe), 670 classes sont éliminées. Finalement, la matrice $\mathbf{X}$ contenant les intensités des pics a pour dimension 253×427. Pour simplifier les descriptions, les classes de pics seront repérées par des numéros et leur position réelle (en Da) sera donnée à la fin pour les pics identifiés comme intéressants.

1.6.1.2 Résultats de l'AG et discrimination

Le modèle de discrimination est construit sur l'échantillon d'apprentissage et sera ensuite testé sur l'échantillon test une fois que le comité final aura été choisi. Afin de rendre les résultats plus fiables et pour vérifier leur stabilité, nous avons réalisé dix répartitions différentes des spectres entre échantillon d'apprentissage et échantillon test.
Nous avons vu que la solution finale pouvait être considérée comme la solution la plus présente dans la population mais aussi comme la solution de la population finale ayant la meilleure fitness. Dans cette application et pour chaque répartition, ces deux critères sont confondus car la solution la plus présente est toujours la meilleure de la population. Nous allons étudier la convergence de l'AG en étudiant la structure des populations finales obtenues.
Tout d'abord, concernant le nombre de pics dans le comité, environ 90% des solutions de chacune des dix populations finales correspondaient à des comités contenant uniquement trois pics. Pour chaque population prise indépendamment, la convergence de ces comités était parfaite, en effet, les trois mêmes pics constituaient l'ensemble des comités à trois pics. Mais ces pics pouvaient être différents pour des répartitions apprentissage/test dis-

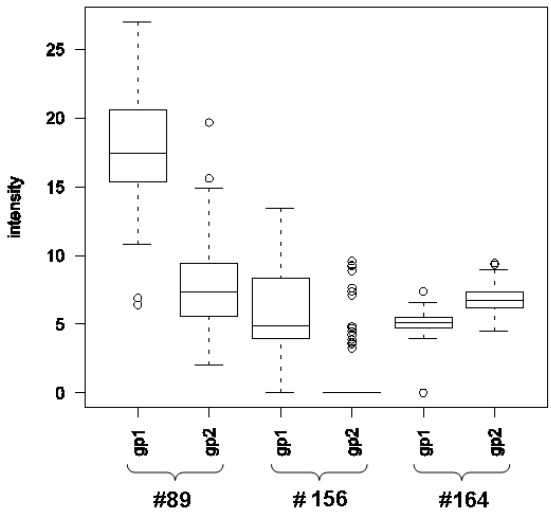
Fig. 1.11 Nombre d'occurences des différents pics trouvés dans les comités pour les dix répartitions apprentissage/test.



tinctes. Pour les seuils d'intensité, la convergence est moins nette car la combinaison la plus fréquente représentait en moyenne 40% des comités à trois pics de la population finale. On peut expliquer ceci notamment par la validation croisée interne qui est effectuée : en changeant légèrement la position d'un seuil, on peut obtenir des résultats légèrement différents qui permettent de mieux prédire certains échantillons tests internes. Les petites variations concernant la position des seuils peuvent donc s'expliquer de cette façon et n'impliquent pas de grands changements dans le modèle global obtenu, ce sont surtout les pics choisis qui importent et ils sont parfaitement stables entre les différentes solutions, ce qui montre leur adaptation à la grande majorité des échantillons tests internes donc leur aspect généralisable. Nous pouvons ensuite étudier les résultats pour la discrimination réalisée en utilisant les comités finaux. Pour les échantillons d'apprentissage, le taux de bien classés obtenu est de 100% pour toutes les répartitions test/apprentissage sauf une où un spectre correspondant à un contrôle est classé comme cancer. Pour les échantillons tests, on atteint un taux de bien classés de 98% (95% pour le groupe contrôle et 99% pour le groupe cancer). Nous avons déjà vu que les comités permettant ces classifications ne comptaient que trois pics. Parmi les dix répartitions effectuées, sept pics différents sont apparus (voir Fig. 1.11) : le pic d'identifiant 89 était toujours présent, quelle que soit la répartition, le pic 164 est apparu dans huit des dix comités et le pic 156 a été utilisé dans la moitié des comités. Les autres pics utilisés étaient plus anecdotiques et peuvent être attribués à des particularités de l'échantillon. De plus, il est intéressant de noter que les valeurs des seuils pour les pics sélectionnés ne varient pas beaucoup d'une répartition à l'autre. Par exemple, pour le pic 89, le seuil est situé entre 11.13 et 12.71 alors que l'intensité de ce pic varie entre 2 et 27. Ainsi, on peut dire que la discrimination est efficace et que les résultats sont reproductibles.

Etudions maintenant les pics retenus le plus fréquemment, c'est-à-dire les pics d'identifiants 89, 164 et 156. Leurs positions respectives sont 245 Da, 434 Da et 649 Da. Leur intensité pour les différents groupes est présentée dans la Fig. 1.12. On peut noter que le pic 156 est absent pour la majorité des spectres du groupe 2 (la valeur zéro est la plus souvent

Fig. 1.12 Boxplots des intensités des pics d'identifiants 89, 164 et 156 avec une boîte par groupe.



rencontrée). C'est pourquoi le boxplot n'est constitué que d'un segment et de points isolés qui ne représentent qu'une faible partie de la distribution. Pour les trois pics, les valeurs d'intensité sont réellement différentes entre les groupes, l'AG a donc bien trouvé des pics capables de réaliser une bonne discrimination des groupes. De plus, on peut constater que les pics trouvés ici ont déjà été mis en évidence par d'autres études. Le pic 89 a été identifié par Alexe *et al.* (2004) et Sorace & Zhan (2003), le pic 164 par Alexe *et al.* (2004); Sorace & Zhan (2003); Zhu *et al.* (2003) et Jeffries (2004). Enfin le pic 156 a été trouvé par Jeffries (2004). Attendu que ces auteurs ont tous utilisé des méthodes différentes d'extraction des pics et de sélection, le fait de trouver des résultats similaires tend à montrer que les pics sélectionnés ont une réelle signification. On peut donc conclure que notre méthode donne de bons résultats pour deux groupes et fournit de plus une combinaison originale de pics qui n'avaient pas encore été trouvée.

1.6.2 Données neurologiques

1.6.2.1 Description de la population

Dans le cadre du système français de surveillance de la maladie de Creutzfeld-Jacob (notée CJD), l'analyse de Western Blotting CSF 14-3 3 et la collecte d'échantillons de sang ont été réalisées pour environ 8000 patients initialement suspectés de CJD (la période de surveillance s'est étendue du 01/01/1996 au 01/04/2005). Les échantillons ont été collectés et centrifugés selon les protocoles classiques puis congelés jusqu'à analyse. Finalement, dix patients pour lesquels le diagnostique de CJD a été confirmé, dix patients pour lesquels on a finalement diagnostiqué la maladie d'Alzheimer et 17 patients souffrant de désordres psychiatriques, ont été inclus dans cette étude. Nous sommes donc en présence de trois groupes, ce qui nous conduit à utiliser l'analyse par paires de groupes.

1.6.2.2 Les données

Pour générer les données de SELDI, 5 μL de chaque échantillon de sérum ont été dilués dans 7.5 μL d'urée à 8 M, 1% de CHAPS (détergent) et agités. Ensuite, 5 μL de cet échantillon dénaturé ont été mélangés à 195 μL d'un tampon de liaison dépendant du type de surface utilisé. Dans cette étude, quatre types de surfaces ont été utilisés : Q10 (échange d'anions faibles) à pH 9, CM10 (échanges de cations forts) à pH 4, H50 (chromatographie en phase inverse) à 10% d'acetonitrile et IMAC30 chargé avec du nickel. Les barrettes ont été assemblées dans un bioprocesseur et par deux fois, 100 μL des échantillons dénaturés et dilués ont été incubés pendant une heure sur un agitateur à température ambiante. Les barrettes ont été rincées trois fois avec le tampon de lavage approprié et un bref rinçage à l'eau a été réalisé. Les barrettes sont extraites du bioprocesseur et séchées à l'air. 0.8 μL d'une solution d'acide sinapinique saturé est appliqué deux fois sur chaque spot avant un séchage à l'air.
L'analyse SELDI a été réalisée dans un lecteur PBS-II ProteinChip (Ciphergen Biosystems). Pour une même expérience comparative, l'obtention des données est réalisée dans les mêmes conditions de calibration, de FOCUSING MASS, d'intensité du laser et de sensibilité du détecteur. Chaque spectre est la moyenne d'au moins 65 tirs de lasers et est calibré de façon externe avec le All-in-1 Protein Standard II (Ciphergen Biosystems). L'obtention des spectres est finalement réalisée dans le logiciel ProteinChip version 3.2 (Ciphergen Biosystems).

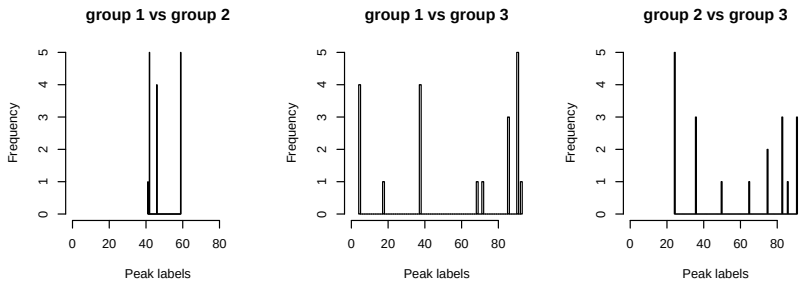
1.6.2.3 Prétraitement

Les étapes de prétraitement décrites précédemment sont appliquées. Tout d'abord, les pics sont extraits dans chaque spectre, puis le seuil pour la hauteur de vallée est choisi et appliqué. En moyenne, 65 pics sont extraits dans chaque spectre. Enfin, les pics sont alignés entre les spectres ce qui conduit à retenir 93 classes de pics. On obtient donc une matrice $\mathbf{X}$ des intensités de dimensions 37×93.

1.6.2.4 Résultats de l'AG et discrimination

Comme les données comportent $k = 3$ groupes, il y a $k(k-1)/2 = 3$ comparaisons deux à deux possibles et donc trois populations évoluent parallèlement dans l'AG. A la fin, on obtient donc trois populations finales donc trois comités qui nous permettent à la fois de connaître les pics caractéristiques de chaque comparaison de deux groupes et d'être capables de calculer un taux de bien classés global. Ainsi, si un pic apparaît dans plusieurs comités, on sait que son pouvoir de discrimination s'étend à plus de deux groupes. L'information obtenue à la fin de l'AG est donc très complète.
Ce jeu de données est très petit (seulement 37 spectres), cela ne nous permet pas de faire de séparation apprentissage/test *a priori* de l'échantillon de départ. Cependant, pour évaluer la reproductibilité des résultats de l'AG nous l'avons appliqué cinq fois de suite à l'ensemble des spectres. L'AG étant une méthode stochastique, il serait tout à fait possible d'obtenir des résultats tout à fait différents si l'algorithme *tombait* successivement dans différents optima locaux. Finalement, les pics sélectionnés au cours des cinq applications sont représentés dans la Fig. 1.13. Pour la première comparaison (Alzheimer contre CJD), les comités étaient constitués de trois pics : les pics d'identifiants 42 et 59 ont toujours été sélectionnés et le pic 46 a été trouvé dans quatre des cinq répétitions (remplacé par le pic 41 dans la dernière). Concernant la deuxième comparaison (Alzheimer contre Psychiatrique), les comités étaient composés de quatre pics : le pic 91 a toujours été sélectionné et les pics 4 et 38 ont été

Fig. 1.13 Histogrammes des fréquences de sélection des pics.



trouvés quatre fois sur cinq. La dernière comparaison (CJD contre Psychiatrique) a requis quatre pics : le pic d'identifiant 24 a été sélectionné les cinq fois et trois pics (83, 91 et 36) ont été trouvés trois fois.

Ainsi, les meilleurs résultats sont obtenus pour la première comparaison pour laquelle les comités sont très stables, la deuxième comparaison donne également des résultats satisfaisants mais c'est un peu moins précis pour la dernière. En effet, d'autres résultats ont montré que le premier groupe se distingue facilement des deux autres et spécialement du second. Ceci peut certainement s'expliquer par le fait que le premier groupe contient plus de spectres que les autres et donc, que les résultats le concernant sont plus fiables.
Nous avons déjà évoqué que la faible taille de l'échantillon ne nous a pas permis de réaliser une vraie validation croisée *externe*. Ceci est d'autant plus vrai que nous avons besoin d'une taille suffisante pour pouvoir réaliser la validation croisée *interne* qui est tout à fait indispensable et qui nécessite de mettre de côté 30% des échantillons à chaque génération. Il est donc impossible, notamment pour les deux derniers groupes qui contiennent seulement dix spectres, de diviser l'échantillon de départ avant de réaliser l'AG.
Cependant, pour tester un peu mieux la fiabilité de nos résultats, nous avons choisi les meilleurs comités par AG appliqué à tous les spectres puis les règles de décision ont été fixées et testées par une validation croisée sur cinq groupes (notée 5-FCV). Ainsi, les spectres utilisés sont différents à chacune des cinq étapes ainsi que les proportions de chacun des groupes par rapport aux seuils ce qui conduit à différentes règles de décision. Ceci est évident moins strict qu'une vraie validation croisée mais c'est le mieux que l'on puisse faire sur ces données. La 5-FCV a été réalisée pour chacune des cinq répétitions. Les résultats de discrimination obtenus sont donnés dans la Tab. 1.2 qui montre qu'en moyenne, on arrive à 98.4% de spectres bien classés.

Enfin, on peut remarquer que les pics trouvés ici n'avaient pas été identifiés en utilisant les méthodes statistiques univariées que l'on utilise classiquement sur ce type de données et qui sont proposées dans le logiciel ProteinChip. Ceci est essentiellement dû au fait que certains pics peuvent être inutiles pour discriminer les groupes en eux-mêmes mais, combinés à d'autres pics, ils deviennent très efficaces. Aucun des pics pris indépendamment n'aurait pu fournir une aussi bonne discrimination que celle obtenue ici et c'est ce qui se passe dans la plupart des jeux de données, d'où l'intérêt de construire une méthode multivariée. Rappelons également que les tests univariés utilisés se basent sur les intensités brutes (non discrétisées), ce qui est certes plus précis, mais également moins robuste car si on change légèrement

Tab. 1.2 Nombre de mauvaises classifications par 5-FCV pour chacune des cinq répétitions de l'AG.

répétition	nombre d'erreurs
1	1
2	0
3	2
4	0
5	0

l'échantillon, les résultats obtenus seront très différents.

1.6.3 Bilan sur l'utilisation des AG pour la recherche de biomarqueurs en SELDI-TOF

La spectrométrie de masse SELDI-TOF est une technologie intéressante à plus d'un titre (rapidité de l'obtention des résultats, diversité des surfaces d'échanges, coûts modérés,...) mais elle souffre d'un manque de précision, spécialement en intensité. Ceci rend son aspect quantitatif difficilement exploitable. Dans ce contexte, la discrétisation des données d'intensité par l'utilisation d'un seuil semble être un bon moyen d'utiliser tout de même cette information mais sans trop *compter* sur des données extrêmement variables. De plus, les données n'étant généralement pas très nombreuses, le recours des arbres de classification classiques n'étaient pas adapté car ils utilisent rapidement trop peu d'information, conduisant à du surajustement. D'où la mise en place de la méthode de *forêt de branches*.
Cependant, viennent ensuite les problèmes du choix des pics à utiliser pour discriminer (les biomarqueurs potentiels) et du choix des seuils servant à la discrétisation. Les nombres de pics et de seuils potentiels étant assez importants, le nombre de possibilités devient vite impossible à explorer de façon exhaustive. Les AG ont alors tout à fait leur place pour la résolution de ce type de problèmes et l'utilisation d'un jeu de données classique et abondamment utilisé a permis de mettre en évidence l'efficacité de la sélection effectuée par l'AG mis en place.
Pour ce qui concerne les données à plus de deux groupes, il n'y a théoriquement pas de limite au nombre de groupes possible, la seule limite vient du temps de calcul qui augmente quand on rajoute des comparaisons. Cependant, l'étape de sélection étant commune à l'ensemble des populations, le temps de calcul ne triple pas pour trois groupes par exemple, il est légèrement accru mais ne double même pas. Par exemple, l'AG a été programmé sous Matlab© et utilisé avec un modèle basique de PC et l'algorithme pour trois groupes prend 8 minutes pour 500 générations et une population de 200 individus.
Enfin, les résultats obtenus sont satisfaisants pour les deux jeux de données tant pour la pertinence des pics sélectionnés que pour leur faible nombre : on réalise une discrimination de bonne qualité en utilisant peu de pics, ce qui était notre objectif.

1.7 Autre application : la recherche d'intervalles discriminants en NIRS

En chimiométrie, la spectrométrie proche infra-rouge (notée SPIR) est une technique très abondamment utilisée qui fournit des données très complètes (aussi bien physiques que chimiques) en un temps très court. Cependant, une partie de l'information recueillie est due à du bruit, par exemple, au signal de la matrice qui contient l'échantillon, et n'a donc aucun intérêt pour l'étude de l'échantillon lui-même. De plus, suivant la problématique rencontrée, on va s'intéresser à une partie précise de l'information concernant l'échantillon. Dans ce contexte, extraire l'information d'intérêt à l'intérieur des spectres obtenus est tout à fait utile. En particulier, Thomas (1994) a montré que la sélection de variables était une bonne méthode pour améliorer les capacités prédictives en se débarrassant de l'information inutile. En SPIR, les variables sont les différentes longueurs d'onde pour lesquelles on a mesuré le spectre.
Une fois de plus, nous nous sommes placés dans un contexte de discrimination de spectres et plus précisément sur un jeu de données *a priori* divisé en quatre groupes. Il s'agit d'échantillons de mélanges de tabacs qui ont été classés en fonction de leur composition chimique (Figuères *et al.*, 2004). Dans ce cas, utiliser le spectre entier sans sélection de variables conduit à surajuster l'échantillon d'apprentissage. Nous avons donc décidé de mettre au point un AG qui permettrait de sélectionner les longueurs d'onde les plus à même de réaliser la discrimination des groupes. De plus, comme les variables adjacentes sont très fortement corrélées (car chimiquement liées), nous avons choisi d'extraire des intervalles de longueurs d'onde adjacentes et non des longueurs d'onde isolées comme cela est réalisé la plupart du temps. En effet, si une longueur d'onde est intéressante pour la discrimination, les longueurs d'onde voisines ont de fortes chances de l'être aussi. Nous appellerons la méthode obtenue AG-AFD.

1.7.1 Les étapes de AG-AFD

Codage des solutions
Dans cette application, une solution correspond à un ensemble de longueurs d'onde réparties en intervalles, on peut donc les repérer par les bornes de chaque intervalle. De plus, pour garder une taille de codage constante, comme pour le SELDI, on définit une taille maximale qui correspond à un nombre maximum d'intervalles, N_m (ici aussi la parcimonie sera recherchée et cette borne n'est donc pas un problème). Le vecteur représentant une solution contiendra successivement, le début et la fin du premier intervalle, puis du deuxième,... et on finit par les NA pour compléter si nécessaire.

Population initiale
Nous n'avons toujours pas d'information *a priori* sur les solutions, on génère donc une population initiale aussi hétérogène que possible en choisissant au hasard le nombre d'intervalles entre 1 et N_m. Ensuite, on choisit les bornes des intervalles en respectant une contrainte de longueur minimale pour un intervalle et d'espace minimal entre deux intervalles successifs.

Opérateur de mutation
A nouveau, il faut que l'opérateur de mutation autorise tous les changements possibles de façon à pouvoir passer de n'importe quelle solution à n'importe quelle autre en un nombre

fini d'étapes. On envisage donc, comme précédemment, trois cas : la suppression d'un intervalle, l'ajout d'un intervalle et le déplacement des bornes d'un intervalle déjà présent. Les probabilités de chacun des types de mutation et les modalités d'application de l'opérateur sont définies exactement comme pour le SELDI (cf. paragraphe 1.5.3).

Opérateur de croisement
Le codage étant très proche de celui des solutions dans le cas du SELDI, l'opérateur de croisement est appliqué strictement de la même façon.

Opérateur de sélection
La première étape pour pouvoir appliquer la sélection est bien entendu de définir la fonction de fitness à optimiser. Comme nous l'avons vu, l'objectif principal est la discrimination. Dans ce cas, cela va être beaucoup plus simple que pour le SELDI. En effet, les données issues du SPIR sont beaucoup plus fiables, il n'est donc pas nécessaire d'introduire une méthode spéciale pour prendre en compte la variabilité des données, nous allons donc simplement appliquer une Analyse Factorielle Discriminante au sens de Fisher (notée AFD) (Mardia *et al.*, 1979). En général, les méthodes portant sur des données spectrales ont plutôt tendance à utiliser PLS pour réaliser la discrimination car l'AFD est sensible aux problèmes mal dimensionnés (plus de variables que d'individus) et à l'auto-corrélation des variables (Naes & Mevik, 2001). Cependant, dans notre problématique, nous réalisons de la sélection de variables qui, si elle est efficace, devrait conduire à un nombre de variables inférieur au nombre d'individus. De plus, de manière générale, l'AFD est une meilleure méthode pour faire de la discrimination que PLS (Barker & Rayens, 2003) car elle prend en compte à la fois la variabilité inter et intra groupes alors que PLS n'a pas pour objet de minimiser les probabilités de mauvaises classifications. Par ailleurs, pour éviter le surajustement, nous mettrons à nouveau une partie de l'échantillon de côté afin de tester la qualité du modèle.
Dans la fitness, nous voulons aussi tenir compte de la parcimonie du modèle choisi. Pour cela, nous allons introduire dans le critère une fonction du nombre d'intervalles choisis et une fonction du nombre de longueur d'ondes choisies. Les deux doivent être utilisés pour pénaliser le cas où un seul intervalle couvrirait la quasi totalité du spectre (ce qui pourrait arriver si on ne tenait compte que du nombre d'intervalles) et le cas où de nombreux intervalles très courts seraient utilisés (ce qui ne serait pas pénalisé si on ne considérait que le nombre de longueurs d'onde). Finalement, chaque terme est ramené dans l'intervalle $[0, 1]$ et la fitness peut être défini comme suit :

$$fitness = \alpha_1 \times \tau + \beta_1 + \alpha_2 \times \left(1 - \frac{nb_i}{N_m}\right) + \beta_2 + \alpha_3 \times \left(1 - \frac{nb_{lo}}{N_t}\right) + \beta_3,$$

où τ est le pourcentage de bien classés, nb_i est le nombre d'intervalles sélectionnés, nb_{lo} est le nombre de longueurs d'onde sélectionnées et N_t est le nombre total de longueurs d'onde du spectre (on rappelle que N_m est le nombre maximal d'intervalles autorisé).

Une fois que la fitness est défini, on le calcule pour l'ensemble des individus de la population courante et on réalise l'étape de sélection exactement de la même façon que pour le SELDI.

Tab. 1.3 Distribution des spectres dans les deux échantillons et les quatre groupes chimiques.

échantillon	groupe 1	groupe 2	groupe 3	groupe 4
apprentissage	26	48	47	90
test	8	51	11	82

1.7.2 Application

Nous disposions d'un échantillon de 363 spectres qui avaient été répartis en deux sous-échantillons : 211 pour l'échantillon d'apprentissage et 152 pour l'échantillon test. Le nombre de spectres appartenant à chacun des quatre groupes chimiques pour les deux échantillons est donné dans la Tab. 1.3. GA-AFD a été programmé sous R© et appliqué à l'échantillon d'apprentissage.

On peut alors étudier la convergence empirique de l'algorithme. On peut tout d'abord considérer l'évolution des termes du critère le long des générations. Cette évolution est représentée dans la Fig. 1.14. La valeur représentée pour chaque étape est la moyenne sur l'ensemble de la population. Le graphique montre que le taux de bien classés augmente au cours des générations. Cette évolution n'est pas régulière à cause de la validation croisée interne (l'échantillon test interne change à chaque génération). Pour la proportion du spectre utilisée, elle décroît régulièrement au cours du temps et finit par se stabiliser. Quant au nombre d'intervalles, il est plus long à se stabiliser mais il finit par se situer autour de quatre. On peut remarquer une irrégularité à la fin de l'algorithme, ceci est simplement dû au fait que l'on réaugmente le taux de mutation et donc que la population redevient plus hétérogène.

Ensuite, on peut également observer la convergence de l'algorithme en considérant la composition de la population finale. La Fig. 1.15 (partie de gauche) montre que la plupart des solutions (71%) ont quatre intervalles. Puis, pour les individus ayant le nombre d'intervalles le plus fréquent, on observe les intervalles obtenus. Ces intervalles sont représentés dans la partie droite de la Fig. 1.15 avec une ligne par solution. On s'aperçoit bien que les intervalles trouvés sont très proches. De plus, les exceptions que l'on trouve (tant pour le nombre que pour la position des intervalles) sont essentiellement dues à la réaugmentation du taux de mutation. Mais le fait que la population soit toujours très homogène montre une certaine stabilité de l'optimum trouvé et on a donc moins de risque d'avoir obtenu un optimum local. Finalement, on retiendra les intervalles [856,874], [2268,2294], [2332,2342] et [2478,2488] (en nm).

Afin de vérifier la pertinence des intervalles trouvés, nous avons réalisé l'AFD de ces longueurs d'onde pour l'échantillon d'apprentissage et pour l'échantillon test. Nous avons également comparé les résultats avec ceux obtenus par d'autres méthodes : AFD appliquée à l'ensemble du spectre, k-plus proches voisins (noté k-ppv) (Hastie *et al.*, 2001), arbres de classification (Hastie *et al.*, 2001), PLS suivi par une sélection de variables par VIP (*Variable Importance in Prediction*) puis application de l'AFD sur les variables retenues (notée PLS-DA + VIP) (Tenenhaus, 1998). Le nombre de voisins et le nombre de composantes de PLS ont été optimisés par validation croisée. Enfin, deux AG décrits dans Leardi (2000) qui

Fig. 1.14 Représentation de la convergence du critère durant AG-AFD (∘ pour le taux de bien classés, + pour le nombre d'intervalles et □ pour le nombre de longueurs d'onde).

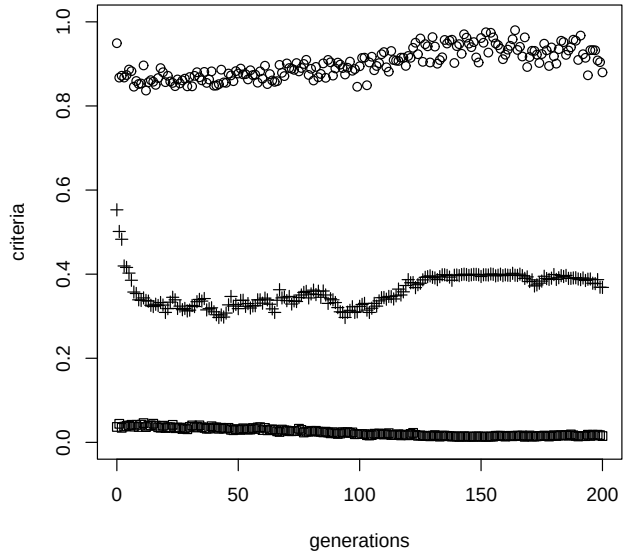
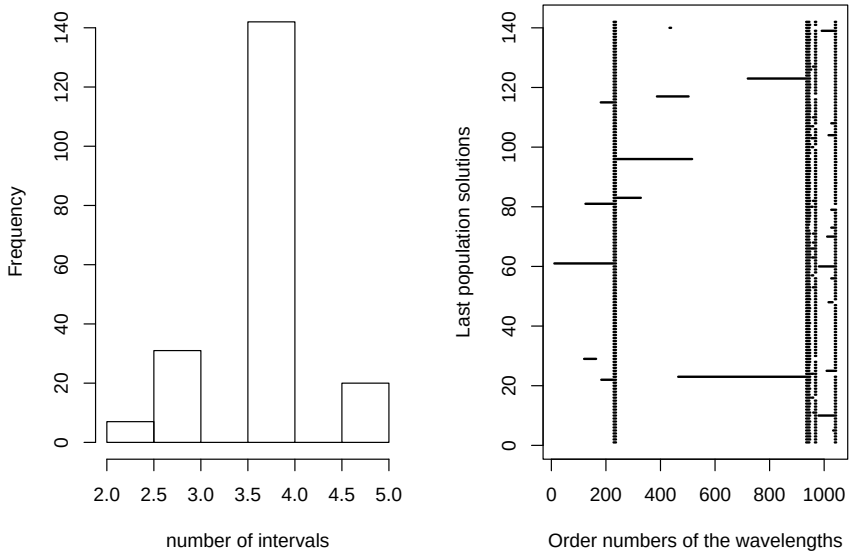


Fig. 1.15 Représentation de la convergence de la dernière population obtenue par AG-AFD : distribution du nombre d'intervalles dans la population finale (partie de gauche) et représentation des intervalles de ces solutions (partie de droite).



Tab. 1.4 Résultats pour différentes méthodes de discrimination : proportion de spectres bien classés pour l'échantillon d'apprentissage et l'échantillon test.

Méthode	Nb de longueurs d'onde utilisées	Echantillon d'apprentissage	Echantillon test
AFD	1050	100	86
arbres	1050	89	64
k-ppv ($k = 3$)	1050	87	70
PLS-DA + VIP	170	79	84
AG-PLS (1)	48	98	94
AG-PLS (2)	85	98	95
FDA-GA	36	97	94

combinent les AG et PLS ont été utilisés (notés AG-PLS(1) et AG-PLS(2)). Les résultats sont présentés dans la Tab. 1.4.
Pour ce qui est de l'échantillon d'apprentissage, AG-AFD et AFD ont donné de très bons résultats. Quant à l'échantillon test, les meilleurs taux de bien classés sont obtenus par AG-PLS(1), AG-PLS(2) et AG-AFD avec des résultats très proches. Mais on peut considérer qu'AG-AFD est plus efficace car le nombre de longueurs d'onde utilisées est plus faible (seulement 3.4% du spectre contre 4.6% pour AG-PLS(1) et 8.1% pour AG-PLS(2)), il semble donc qu'AG-AFD a extrait une information très pertinente tout en évitant le sur-ajustement. Pour les autres méthodes, les arbres et k-ppv fournissent des résultats très moyens notamment par rapport à l'AFD appliquée à l'ensemble du spectre. Enfin, PLS-DA + VIP utilise 16% des longueurs d'onde, les résultats sont proches de ceux de l'AFD pour l'échantillon test et sont assez médiocres pour l'échantillon d'apprentissage. Ainsi, cette méthode évite le sur-ajustement mais les résultats sont moins précis que ceux de AG-AFD alors qu'elle utilise plus d'information, la sélection de variables par VIP est donc moins efficace que celle de l'AG-AFD. Globalement, les trois méthodes incluant PLS sont assez satisfaisantes pour la discrimination de l'échantillon test mais la sélection de variables permise par AG-AFD semble être plus intéressante.

1.7.3 Conclusion

Globalement, on constate à nouveau une efficacité de l'utilisation des AG pour la sélection de variables et surtout leur grande flexibilité. Ils permettent d'introduire facilement des contraintes, soit dans la construction des opérateurs (pour imposer une certaine longueur aux intervalles par exemple), soit sous forme de pénalité dans la fitness global (ici, pour favoriser la parcimonie). Ce type de méthode est donc totalement adapté à l'utilisation de la spectrométrie qui fournit des données en très grand nombre, avec des informations plus ou moins intéressantes et des caractéristiques propres à chaque technique.

Chapitre 2

L'alignement de gels d'électrophorèse en deux dimensions

2.1 Un problème simple... en apparence

L'électrophorèse bidimensionnelle (ou électrophorèse 2D) est une technique de séparation des protéines d'un échantillon. Elle a été introduite par Klose (1975) et O'Farrell (1975). Elle permet d'avoir une vision d'ensemble de l'état protéique d'un tissu pour des conditions et à un moment donnés. La séparation des protéines se fait selon deux critères : leur poids et leur point iso-électrique (lié à leur charge électrique). Ces deux grandeurs sont caractéristiques des protéines qui doivent donc s'immobiliser à un point précis du gel. Afin de pouvoir repérer les protéines dans le gel, on introduit un colorant et le gel obtenu, une fois la migration terminée, est scanné.
En théorie, une même protéine, placée sur deux gels possédant les mêmes propriétés et manipulés selon le même protocole, est censée se retrouver strictement à la même place. On peut ainsi espérer, pour des conditions biologiques distinctes (évolution dans le temps, différents états physiologiques,...), repérer les protéines qui apparaissent et disparaissent. De plus, le colorant est choisi de sorte à ce qu'il soit possible de relier l'intensité de la coloration à la quantité de protéine présente sur le gel. Donc, outre les apparitions/disparitions, on peut espérer repérer des changements de la quantité de protéines.
Cependant, comme dans toute expérimentation biologique, un certain nombre de facteurs plus ou moins bien maîtrisés varient entre deux manipulations (Salmi *et al.*, 2002; Voss & Haberl, 2000). Les principales sources de variabilité sont les suivantes :

- la structure du milieu : les protéines évoluent dans un gel de polyacrylamide. C'est pourquoi l'électrophorèse 2D est souvent appelée 2D-PAGE pour *2D Poly Acrylamide Gel Electrophoresis*. Ce gel n'est pas forcément tout à fait homogène et contient un gradient de pH (pour la séparation en fonction des points isoélectriques) qui peut comporter des différences, même à l'intérieur d'un lot.
- les caractéristiques de l'échantillon qui contient les protéines (notamment sa concentration en sels),
- la nature du champ électrique (utilisé pour la séparation en fonction des points isoélectriques).

Il en résulte une distorsion des gels qui empêche leur superposition directe. C'est pourquoi il est nécessaire de construire une méthode capable d'aligner les gels, c'est-à-dire d'établir les correspondances entre les protéines détectées, appelées *spots*.
Si on avait simplement un problème de distorsion entre deux gels contenant la même infor-

mation, le problème serait simple à résoudre. Des algorithmes existants tels que Procuste (Green, 1952; Hurley & Cattell, 1962) seraient tout à fait capables de trouver la (ou les) transformation(s) à effectuer. Mais, dans le contexte des gels 2D, le but est d'étudier des conditions biologiques différentes, et dans ce cas, on n'a aucune raison de retrouver exactement les mêmes protéines, donc les mêmes spots dans les différents gels. On doit donc aligner des listes de spots sans savoir qui est apparié avec qui ni même si tel spot est apparié avec un quelconque spot dans un autre gel. Sachant qu'on peut avoir plus de mille spots sur un gel, le nombre de possibilités d'appariements est tout à fait colossal (d'autant plus si on a de nombreux gels).

De plus, il est impossible de considérer une transformation simple qui reflète la distorsion globale du gel. La nature des facteurs de la déformation fait qu'elle n'est pas homogène. Ainsi, même dans le cas simple où on connaîtrait les appariements, appliquer une méthode d'alignement globale (par exemple, la méthode Procuste) serait inadapté. Il va donc falloir diviser le gel en cellules à l'intérieur desquelles on peut approcher la distorsion par une transformation locale simple.

Il existe deux grandes familles de méthodes pour aligner des gels : des méthodes directes qui travaillent sur l'image brute et des méthodes qui utilisent des caractéristiques de l'image préalablement extraites. Cependant, l'étape d'extraction des caractéristiques (typiquement la détection des spots) est souvent susceptible de faire des erreurs et donc de fournir à l'algorithme d'alignement des données initiales de mauvaise qualité.

Par contre, l'alignement direct des images brutes est souvent très coûteux en calculs et donc en temps. Une fois encore, le développement des capacités de calcul des ordinateurs a finalement permis l'émergence de telles méthodes (Smilansky, 2001; Veeser *et al.*, 2001). Notons toutefois que ces méthodes sont souvent des hybrides où les images brutes sont utilisées pour estimer les paramètres de transformation globale.

Par exemple, dans Wang & Feng (2005), une décomposition par ondelettes est utilisée pour corriger les déformations globales et estimer les paramètres d'une transformation affine. Ensuite, des points sont automatiquement sélectionnés dans l'image pour corriger les distorsions locales et améliorer la précision de la solution. Les ondelettes sont également utilisées dans Kaczmarek *et al.* (2004) pour débruiter l'image.

Marengo *et al.* (2003) utilisent une technique intermédiaire de logique floue qui réalise une binarisation de l'image de départ, par application d'un seuil d'intensité censé séparer le bruit de fond du signal. Ensuite, les zones où un signal a été détecté sont divisées en cellules élémentaires dans lesquelles la probabilité d'avoir un spot est modélisée. Ces probabilités pour chaque cellule et dans chaque gel deviennent les nouvelles variables auxquelles on applique une Analyse en Composantes Principales pour identifier des zones communes ou différentes entre gels.

Cependant, l'étape d'extraction des caractéristiques comporte un intérêt majeur : distinguer l'information intéressante du bruit au sens large alors que dans les méthodes directes toute l'information est prise en compte sans distinction. En outre, les algorithmes de détection de spots sont de plus en plus performants. Des méthodes permettant de modéliser la forme des spots et ainsi de limiter les erreurs de détection ont été mises au point (Rogers *et al.*, 2003). On peut tout de même noter que les biologistes continuent souvent à effectuer des corrections manuelles après détection automatique.

Pour corriger les déformations globales du gel, certains modélisent directement les phénomènes physico-chimiques à l'origine de ces distorsions (Gustafsson *et al.*, 2002). Pour les autres méthodes, la plupart utilise les coordonnées des spots détectés et recourt à des spots de

repère appelés *landmarks*. Il s'agit de spots pour lesquels on connaît les spots correspondants dans les différents gels et qui servent de points d'ancrage pour réaliser les transformations. On va alors décomposer l'image en cellules, des rectangles par exemple (Potra & Liu, 2006), dans lesquelles on va supposer que la transformation est plus simple, et ainsi estimer cette transformation grâce aux landmarks.
Enfin, la quasi-totalité des algorithmes proposés repose sur une comparaison deux à deux des gels. La plupart du temps, si on a plus de deux gels, un gel est choisi comme *gel de référence*, puis les autres sont alignés sur lui. Dans un tel cas, si un spot est présent dans plusieurs gels mais pas dans le gel de référence, il sera impossible de le repérer. Le choix du gel de référence est donc crucial et influe beaucoup sur les résultats s'il existe des différences importantes entre gels (ce que l'on cherche généralement à mettre en évidence !). Quelques alternatives ont été proposées, notamment la construction d'un gel *idéal* (Potra & Liu (2006)) à partir de l'information de l'ensemble des gels à aligner auquel sera ensuite aligné chaque gel.
En ce qui nous concerne, nous avons décidé, en accord avec les biologistes, de travailler sur des gels pour lesquels les spots avaient été automatiquement détectés puis validés par les experts concernés. Pour l'alignement, la seule information que nous prendrons en compte sera donc les coordonnées des spots détectées. En effet, l'alignement pourrait aussi tenir compte des informations concernant la forme des spots mais une telle information est très difficile à caractériser et à quantifier, nous la laisserons donc de côté. Quant à l'intensité, il n'y a pas de raison de l'utiliser dans l'étape d'alignement car l'objectif est d'aligner des gels obtenus dans différentes conditions biologiques et donc pour lesquels les intensités des spots représentant une même molécule sont susceptibles de changer, il ne serait donc pas pertinent de s'en servir. Par contre, l'intensité a une grande importance dans la détection des spots d'intérêt après alignement, c'est-à-dire des spots dont l'intensité change en fonction des conditions (cette étape ne sera pas traitée ici).

2.2 Cas de deux gels : adaptation de Procuste

Dans un premier temps, nous allons considérer l'alignement de deux gels d'electrophorèse 2D seulement. Pour cela, nous allons présenter la méthode Procuste (Green, 1952; Hurley & Cattell, 1962) dans un contexte assez général puis nous verrons comment nous l'utilisons dans notre cas particulier où les correspondances entre spots ne sont pas connues.

2.2.1 La méthode Procuste

Il existe plusieurs formulations pour les méthodes procrustéennes qui, toutes, cherchent à faire correspondre deux configurations décrivant des individus identiques ou appariés en estimant une transformation. Nous allons décrire le problème et sa solution dans le cas précis qui nous intéresse (Schönemann & Carroll, 1970).
On considère deux matrices $\mathbf{X}$ et $\mathbf{Y}$ de même dimension $n \times p$. Chaque ligne représente les coordonnées dans l'espace des p variables de chacun des n individus. Dans le cas en deux dimensions des gels d'électrophorèse, on aura $p = 2$. On suppose que les lignes des matrices sont appariées deux à deux, c'est-à-dire que la première ligne de $\mathbf{X}$ est appariée avec la première ligne de $\mathbf{Y}$, etc,... L'objectif est de trouver une transformation de $\mathbf{Y}$ qui soit le plus exactement superposable avec $\mathbf{X}$. Pour cela, on cherche à minimiser les distances entre les points appariés après transformation. La transformation sera décrite par trois paramètres :

– $\mathbf{T}$, une matrice $p \times p$ de rotation/réflexion,
– s, un scalaire représentant le facteur de dilatation,
– $\mathbf{t}$, un vecteur colonne de longueur p qui représente une translation.

La matrice obtenue par transformation de $\mathbf{Y}$ peut alors s'écrire :

$$\hat{\mathbf{Y}} = s\mathbf{YT} + \mathbf{1}_n\mathbf{t}'.$$

On cherche donc à minimiser la somme des carrés des distances entre points appariés :

$$\begin{aligned} L(s,\mathbf{t},\mathbf{T}) &= tr[(\mathbf{X}-\hat{\mathbf{Y}})'(\mathbf{X}-\hat{\mathbf{Y}})] && (2.1)\\ &= tr[(\mathbf{X}-(s\mathbf{YT}+\mathbf{1}_n\mathbf{t}'))'(\mathbf{X}-(s\mathbf{YT}+\mathbf{1}_n\mathbf{t}'))], && (2.2)\end{aligned}$$

sous la contrainte $\mathbf{T}'\mathbf{T} = \mathbf{I}_p$. Sans cette contrainte, $\mathbf{T}$ ne serait pas une transformation orthogonale, ce serait une transformation linéaire qui ne préserve pas forcément la forme de $\mathbf{Y}$, en particulier, les rapports de distances entre les points de $\mathbf{Y}$.
Pour trouver le vecteur $\mathbf{t}$ optimal, on dérive $L(s,\mathbf{t},\mathbf{T})$ par rapport à $\mathbf{t}$ et on égalise à zéro :

$$\frac{\partial L(s,\mathbf{t},\mathbf{T})}{\partial \mathbf{t}} = 2n\mathbf{t} - 2\mathbf{X}'\mathbf{1}_n + 2s\mathbf{T}'\mathbf{Y}'\mathbf{1}_n = 0,$$

d'où

$$\hat{\mathbf{t}} = \frac{1}{n}(\mathbf{X} - s\mathbf{YT})'\mathbf{1}_n. \quad (2.3)$$

On reporte (2.3) dans (2.1) et on obtient :

$$\begin{aligned} L(s,\mathbf{T}) &= tr[((\mathbf{X}-s\mathbf{YT}) - \frac{\mathbf{1}_n\mathbf{1}_n'}{n}(\mathbf{X}-s\mathbf{YT}))'(\mathbf{X}-s\mathbf{YT}) - \frac{\mathbf{1}_n\mathbf{1}_n'}{n}(\mathbf{X}-s\mathbf{YT}))] && (2.4)\\ &= tr[((\mathbf{I}_n - \frac{\mathbf{1}_n\mathbf{1}_n'}{n})(\mathbf{X}-s\mathbf{YT}))'((\mathbf{I}_n - \frac{\mathbf{1}_n\mathbf{1}_n'}{n})(\mathbf{X}-s\mathbf{YT}))] && (2.5)\\ &= tr[(\mathbf{J_n}\mathbf{X} - s\mathbf{J_n}\mathbf{YT})'(\mathbf{J_n}\mathbf{X} - s\mathbf{J_n}\mathbf{YT})], && (2.6)\end{aligned}$$

où $\mathbf{J}_n$ est la matrice de centrage $\mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n'$.
De la même manière, on dérive $L(s,\mathbf{T})$ par rapport à s et on égalise à zéro :

$$\frac{\partial L(s,\mathbf{t},\mathbf{T})}{\partial s} = 2s\, tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}] - 2\, tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}] = 0$$

d'où

$$\hat{s} = \frac{tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}]}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]}. \quad (2.7)$$

On reporte (2.7) dans (2.6) et on obtient :

$$\begin{aligned} L(\mathbf{T}) &= tr\left[\left(\mathbf{JX} - \frac{tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}]}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]}\mathbf{J}_n\mathbf{YT}\right)'\left(\mathbf{JX} - \frac{tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}]}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]}\mathbf{J}_n\mathbf{YT}\right)\right] && (2.8)\\ &= tr[\mathbf{X}'\mathbf{J}_n\mathbf{X}] + \frac{(tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}])^2}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]} - 2\frac{(tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}])^2}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]} && (2.9)\\ &= tr[\mathbf{X}'\mathbf{J}_n\mathbf{X}] - \frac{(tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}])^2}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]} && (2.10)\end{aligned}$$

Maintenant, minimiser $L(\mathbf{T})$ par rapport à $\mathbf{T}$ sous la contrainte $\mathbf{T}'\mathbf{T} = \mathbf{I}_p$ est équivalent à minimiser $-tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}]$. Pour cela, nous allons utiliser l'inégalité de Kristof (Kristof, 1970) et sa borne inférieure.

Theorème 2.1 *Si* $\mathbf{A}$ *est une matrice diagonale à entrées positives ou nulles et si* $\mathbf{B}$ *est orthogonale, d'après l'inégalité de Kristof :*

$$-tr[\mathbf{BA}] \geq -tr[\mathbf{A}],$$

avec égalité ssi $\mathbf{B} = \mathbf{I}$.

Appliquons cette inégalité à notre problème. On pose $\mathbf{C} = \mathbf{X}'\mathbf{J}_n\mathbf{Y}$. Soit $\mathbf{P\Phi Q}'$ la décomposition en valeurs singulières de $\mathbf{C}$, c'est-à-dire $\mathbf{C} = \mathbf{P\Phi Q}'$. Alors

$$\begin{aligned} -tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}] &= -tr[\mathbf{CT}] && (2.11)\\ &= -tr[\mathbf{P\Phi Q}'\mathbf{T}] && (2.12)\\ &= -tr[\mathbf{TP\Phi Q}'] && (2.13)\\ &= -tr[\mathbf{Q}'\mathbf{TP\Phi}]. && (2.14) \end{aligned}$$

Or, $\mathbf{\Phi}$ est la matrice diagonale des valeurs propres de $\mathbf{C}$ donc $\mathbf{\Phi}$ est bien une matrice diagonale avec entrées positives ou nulles. De plus, d'après la contrainte $\mathbf{T}'\mathbf{T} = \mathbf{I}_p$, $\mathbf{T}$ est orthonormal donc $\mathbf{Q}'\mathbf{TP}$ est orthonormal, on peut donc appliquer l'inégalité de Kristof :

$$-tr[\mathbf{Q}'\mathbf{TP\Phi}] \geq -tr\mathbf{\Phi},$$

avec égalité ssi $\mathbf{Q}'\mathbf{TP} = \mathbf{I}_p$ d'où

$$\mathbf{T} = \mathbf{QP}'. \quad (2.15)$$

Si on résume ce qui précède, les étapes pour estimer les paramètres de la transformation permettant de passer de $\mathbf{Y}$ à $\mathbf{X}$ sont les suivantes :

1. On calcule $\mathbf{C} = \mathbf{X}'\mathbf{J}_n\mathbf{Y}$.
2. On réalise la décomposition en valeurs singulières de $\mathbf{C}$ de sorte que $\mathbf{C} = \mathbf{P\Phi Q}'$.
3. La matrice de rotation optimale est alors $\mathbf{T} = \mathbf{QP}'$.
4. Le facteur de dilatation optimal est alors $s = \frac{tr[\mathbf{X}'\mathbf{J}_n\mathbf{YT}]}{tr[\mathbf{Y}'\mathbf{J}_n\mathbf{Y}]}$.
5. Le vecteur de translation optimal est alors $\mathbf{t} = \frac{1}{n}(\mathbf{X} - s\mathbf{YT})'\mathbf{1}_n$.

2.2.2 Application à l'alignement de deux gels et insertion dans l'AG

Grâce à la solution de Procuste que nous venons de montrer, nous sommes maintenant capables de trouver la transformation $(\mathbf{T}, s, \mathbf{t})$ optimale pour aligner deux listes de points appariés, $\mathbf{X}$ et $\mathbf{Y}$.
Pour revenir aux gels d'électrophorèse 2D, on notera $\mathbf{X}$ et $\mathbf{Y}$ les deux gels. Notons que dans notre application, nous n'avons que deux coordonnées ($p = 2$) et que le vecteur $\mathbf{t}$ sera donc constitué de deux éléments qui sont les paramètres de translation horizontale et verticale. Dans ce cas, on ne connaît les appariements que pour les landmarks, c'est-à-dire pour une toute petite proportion des spots. De plus, on sait qu'il est impossible de trouver une transformation $(\mathbf{T}, s, \mathbf{t})$ unique pour tout le gel. Il va donc falloir trouver des zones considérées comme homogènes pour la transformation et estimer une transformation pour chacune d'elles. Notre méthode se présentera donc en trois étapes :

1. Estimation d'une transformation globale.
2. Recherche de zones homogènes.
3. Estimation d'une transformation par zone.

Nous allons maintenant décrire en détails chacune de ces étapes.

2.2.2.1 Recherche d'une transformation globale

Nous avons expliqué précédemment, qu'une transformation simple globale n'est pas susceptible de permettre un alignement satisfaisant des gels. Cependant, afin de pouvoir trouver des zones homogènes dans lesquelles une transformation simple pourra être estimée, il nous faut obtenir de l'information sur la structure du gel. Le raisonnement est le suivant : supposons qu'on ait construit une transformation globale et qu'on ait trouvé les appariements correspondants. Il est alors possible de calculer, pour chaque appariement, la distance entre les spots appariés après application de la transformation globale. C'est cette distance qui va nous permettre de définir des zones homogènes. En effet, une même transformation (adaptée ou non) appliquée sur une zone homogène, va générer des distances équivalentes en moyenne entre spots appariés. Donc on considérera comme homogène, une zone dans laquelle les distances d'appariement après application de la transformation globale sont comparables. Cette étape sera détaillée dans le paragraphe 2.2.2.2.
De plus, dans toutes les applications que nous avons faites nous avons toujours trouvé un facteur de dilatation compris entre 0.99 et 1. Il a donc une influence négligeable sur la transformation. Nous avons donc décidé, afin d'alléger les calculs, de ne pas rechercher le facteur de dilatation. Nous considérerons donc désormais uniquement les paramètres de rotation et de translation.
Maintenant que nous avons montré l'intérêt de réaliser une transformation globale, voyons comment on procède. Nous allons utiliser la méthode Procuste présentée au paragraphe 2.2.1. Cela consiste à estimer un couple de paramètres *globaux* $(\mathbf{T}_g, \mathbf{t}_g)$, respectivement, la matrice de rotation et le vecteur de translation. Mais pour pouvoir appliquer Procuste, il faut connaître les appariements entre spots. Or, nous connaissons un certain nombre de ces appariements par l'intermédiaire des landmarks. On notera $\mathbf{X}^l$ et $\mathbf{Y}^l$ les coordonnées des landmarks dans les gels $\mathbf{X}$ et $\mathbf{Y}$.

2.2.2.1.1 Recherche d'un critère Les landmarks ne représentent qu'une faible proportion de l'ensemble des spots, or on cherche une transformation satisfaisante pour l'ensemble des spots. Comment quantifier la qualité d'une telle transformation ? Nous avons retenu deux critères :

- la distance entre spots appariés après transformation, qui correspond à une sorte d'erreur (si la transformation était parfaite cette distance serait nulle),
- le nombre de spots appariés, noté a_g.

Une bonne transformation sera ainsi une transformation qui apparie le plus de spots possible en faisant le moins d'erreur possible. En effet, il est simple d'obtenir une très petite erreur si on n'apparie que quelques spots. On peut alors noter que la distance que l'on cherche ici à minimiser est exactement égale au critère optimisé dans Procuste (cf. Eq. 2.1). Notons toutefois qu'il faut légèrement modifier l'écriture de l'Eq. (2.1). En effet, on n'aligne plus $\mathbf{X}$ et $\mathbf{Y}$ mais $\tilde{\mathbf{X}}$ et $\tilde{\mathbf{Y}}$ qui sont les gels réduits aux individus que l'on apparie.

Une fois que l'appariement des spots est réalisé (voir les paragraphes suivants) on peut calculer un critère de qualité que l'on cherche à minimiser :

$$fitness = F(L(\mathbf{t}, \mathbf{T}), a_g). \tag{2.16}$$

Pour la fonction F, nous avons choisi de normaliser chaque terme et de leur appliquer éventuellement un coefficient :

$$F(L(\mathbf{t}, \mathbf{T}), a_g) = \gamma_1(\alpha_1 L(\mathbf{t}, \mathbf{T}) + \beta_1) + \gamma_2(\alpha_2 a_g + \beta_2), \tag{2.17}$$

où $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ sont fixés de sorte à ce que chaque terme soit ramené entre 0 et 1 et (γ_1, γ_2) servent à réaliser un équilibre entre les deux termes. Pour fixer $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ il faut définir des bornes pour $L(\mathbf{t}, \mathbf{T})$ et a_g. Pour $L(\mathbf{t}, \mathbf{T})$, le minimum est évidemment zéro (quand la transformation permet un appariement parfait). Pour le maximum, il faut le fixer, par exemple comme une proportion de la distance maximale entre deux spots sur le même gel. Etant donné qu'on cherche à se rapprocher du minimum de $L(\mathbf{t})$, la borne supérieure n'a pas une grande influence (tant qu'elle reste raisonnable). Pour a_g, si le gel $\mathbf{X}$ a pour dimensions $n_1 \times 2$ et le gel $\mathbf{Y}$, $n_2 \times 2$, alors on a au maximum, $\min(n_1, n_2)$ appariements. Pour le minimum, on peut fixer une limite en dessous de laquelle la pénalisation est toujours maximale, on peut par exemple choisir d'apparier au moins la moitié des spots du plus gros gel. Mais comme pour $L(\mathbf{t}, \mathbf{T})$, cette deuxième borne est moins importante puisqu'on va optimiser de sorte à se trouver près de la première. Quant à γ_1 et γ_2, ils sont fixés empiriquement.

2.2.2.1.2 Appariement pour $(\mathbf{T}_g, \mathbf{t}_g)$ fixés Maintenant que l'on a construit le critère, il faut réaliser l'appariement pour l'ensemble des deux gels. Supposons que l'on ait fixé les valeurs pour $(\mathbf{T}_g, \mathbf{t}_g)$. On applique alors la transformation correspondante au gel $\mathbf{Y}$. On obtient un gel $\hat{\mathbf{Y}}$ dans lequel les coordonnées de chaque spot ont été transformées. Etant donné que l'on souhaite minimiser les distances entre spots appariés, il est maintenant logique d'apparier chaque spot du gel $\mathbf{X}$ avec le spot le plus proche dans $\hat{\mathbf{Y}}$. On note $\mathcal{V}_{\hat{\mathbf{Y}}}(x)$ l'ensemble des spots de $\hat{\mathbf{Y}}$ contenus dans cette fenêtre. On cherche donc, pour chaque spot, $\mathbf{x}$ appartenant au gel $\mathbf{X}$:

$$\arg \min_{\hat{\mathbf{y}} \in \mathcal{V}_{\hat{\mathbf{Y}}}(x)} d(\mathbf{x}, \hat{\mathbf{y}}).$$

On obtient ainsi, une liste de couples de spots appariés. Cependant, à cette étape, on force chaque spot de $\mathbf{X}$ à avoir un correspondant dans $\hat{\mathbf{Y}}$ et on sait que ce n'est pas obligatoirement le cas. Pour éliminer les appariements les plus improbables, on calcule la moyenne, μ, et l'écart-type, σ, des distances entre spots appariés. Soit δ la distance entre deux spots appariés alors, si $\delta - \mu \geq 3\sigma$, on élimine ce couple de la liste des appariements. La dernière étape est alors de recalculer les paramètres de la transformation en appliquant la méthode Procuste aux coordonnées des spots appariés regroupées dans les matrices $\tilde{\mathbf{Y}}$ et $\tilde{\mathbf{X}}$.

2.2.2.1.3 Insertion dans l'AG Dans ce contexte, l'AG va être utilisé pour trouver les paramètres $(\mathbf{T}_g, \mathbf{t}_g)$ qui optimisent le critère introduit dans l'Eq. 2.16. On va donc construire une population de taille T_{pop} dans laquelle chaque individu i ($i = \{1, \ldots, T_{pop}\}$) est représenté par un couple de paramètres $(\mathbf{T}_g^i, \mathbf{t}_g^i)$ qui lui est associé. On peut alors décrire l'AG pour

Fig. 2.1 Pseudo-code de l'algorithme génétique permettant l'alignement global de deux gels.

Initialisation On calcule $(\mathbf{T}_g^l, \mathbf{t}_g^l) = \text{Procuste}(\mathbf{X}^l, \mathbf{Y}^l)$.
On initialise la population de taille T_{pop} avec ces paramètres.
pour $j = \{1, \ldots, N_{gene}\}$ **faire**
    On applique croisement puis mutation pour obtenir de nouveaux paramètres $(\mathbf{T}_g^i, \mathbf{t}_g^i)$ pour $i = \{1, \ldots, T_{pop}\}$.
    On apparie les spots en fonction de ces paramètres pour chaque individu i.
    On recalcule $(\mathbf{T}_g^i, \mathbf{t}_g^i) = \text{Procuste}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$.
    On applique l'opérateur de sélection à la population en utilisant comme fitness, $F(L(\mathbf{t}_g^i, \mathbf{T}_g^i), a_g^i)$.
fin

N_{gene} générations par le pseudo-code de la Fig. 2.1.

L'opérateur de mutation est appliqué avec probabilité $p_m(j)$, qui évolue comme décrit dans le paragraphe 1.3 en fonction de la génération j. Quand on a décidé d'appliquer une mutation à un individu i, on choisit avec équiprobabilité de muter l'un ou l'autre des trois paramètres suivants :

- l'angle de rotation,
- le paramètre de translation horizontale,
- le paramètre de translation verticale.

Ensuite, une fois qu'on a choisi quel paramètre on mute, nous avons décidé de définir un voisinage autour la valeur précédente dans lequel le paramètre sera autorisé à muter. Pour cela, nous générons un nombre suivant une distribution normale centrée autour de l'ancienne valeur du paramètre et dont l'écart-type dépend évidemment du paramètre que l'on mute. Il est intéressant de noter que ces paramètres sont bornés. En effet, d'après les biologistes, il n'est pas raisonnable d'envisager une rotation de plus de 10° d'un côté ou de l'autre, de même, alors que les gels couvrent environ 1500×1500 pixels, une manipulation pour laquelle on trouverait une translation horizontale et/ou verticale de plus de 200 pixels ne pourrait pas être considérée comme valable. L'écart-type des distributions a alors été choisi de sorte à ce qu'on puisse atteindre les 10° ou les 200 pixels en 10 itérations en moyenne. On a donc choisi un écart-type de 1 pour le paramètre de rotation et de 20 pour les paramètres de translation.
Pour l'opérateur de croisement, il est appliqué à la population courante avec une probabilité p_c, constante au cours des générations, comme décrit dans le paragraphe 1.4. Comme chaque individu n'est défini que par trois paramètres, nous avons choisi de ne croiser qu'un seul d'entre eux, choisi avec équiprobabilité.
Quant à la sélection, elle suit la méthode décrite au paragraphe 1.5 en utilisant comme valeur de fitness, $F(L(\mathbf{t}, \mathbf{T}), a_g)$ et en se basant sur les rangs des individus de la population en fonction des valeurs de F. Ces rangs permettent le calcul, pour chaque individu de la population, d'une probabilité de sélection.

A la fin de l'algorithme, on retient la meilleure solution de la population finale et on choisit les paramètres obtenus pour les paramètres de transformation globale, $(\mathbf{T}_g, \mathbf{t}_g)$. Voyons maintenant comment utiliser cette transformation globale pour définir des zones homogènes.

2.2.2.2 Définition de zones homogènes dans les gels

Nous avons vu que la transformation globale allait être utilisée pour observer les distances entre spots appariés après transformation. En effet, on considère que les zones pour lesquelles on trouve des distances comparables sont susceptibles de pouvoir être appariées par une même transformation locale.

2.2.2.2.1 Découpage du gel Il est alors nécessaire de définir des cellules élémentaires dans le gel, dans lesquelles on observera les distances et que l'on cherchera à regrouper pour définir les zones homogènes. Pour cela, nous avons choisi d'utiliser la partition de Voronoï (Fortune, 1987). C'est une décomposition particulière d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace. Définissons cette décomposition pour un espace à deux dimensions (comme c'est le cas pour les gels).
Soit $\mathcal{S}$ un ensemble de points de $\mathbb{R}^2$. Pour adapter à notre contexte, on munira l'espace $\mathbb{R}^2$ de la distance euclidienne usuelle. On pose $\text{card}(\mathcal{S}) = n_c$. Ces n_c points sont appelés *centres*.

Définition 2.1 *On appelle $Vor(c^i)$, la cellule de Voronoï associée au point $c^i \in \mathcal{S}$, c'est-à-dire l'ensemble des points de $\mathbb{R}^2$ qui sont plus proches de c^i que de n'importe quel autre point de $\mathcal{S}$:*

$$Vor(c^i) = \{x \in \mathbb{R}^2 : \forall p \in \mathcal{S}\backslash\{c^i\},\ d(x, c^i) < d(x, p)\},$$

où $d(x, c^i)$ est la distance euclidienne entre les points x et c^i.

En fait, pour construire la partition de Voronoï, il faut commencer par construire des frontières entre les points de $\mathcal{S}$. Pour définir ces frontières, considérons tout d'abord deux points, c^i et c^j de $\mathcal{S}$. L'ensemble des points de $\mathbb{R}^2$ qui sont à la même distance de c^i et de c^j sont les points de la médiatrice du segment $[c^i, c^j]$. Donc, les points de $\mathbb{R}^2$ qui sont plus proches de c^i que de c^j sont les points appartenant au demi-plan $H(c^i, c^j)$ délimité par la médiatrice de $[c^i, c^j]$ et contenant c_i. Ainsi, pour construire la cellule de Voronoï $\text{Vor}(c^i)$ qui contient tous les points plus proches de c^i que de tout autre point de $\mathcal{S}$, il suffit de réaliser l'intersection de tels demi-plans :

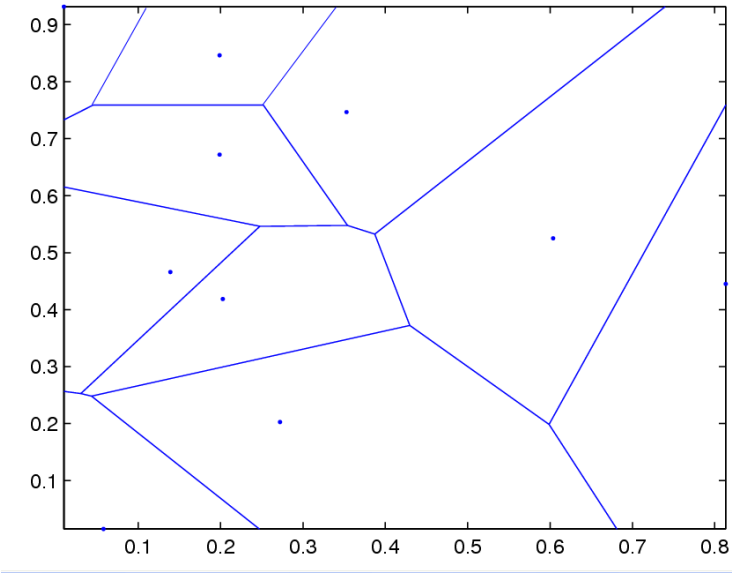
$$\text{Vor}(c^i) = \bigcap_{j \neq i} H(c^i, c^j).$$

Il est intéressant de noter que les cellules obtenues sont des polygones convexes. On obtient donc une partition de $\mathbb{R}^2$ comme illustré par la Fig. 2.2.

Il est maintenant possible de réaliser une partition des gels en prenant comme ensemble $\mathcal{S}$ l'ensemble des landmarks dans chacun des gels. Ce choix permet d'avoir une partition comparable dans les deux gels.

2.2.2.2.2 Rassemblement des cellules Une fois la partition réalisée, on pourrait directement réaliser une transformation par cellule. Si on a assez de landmarks, on obtient de petites cellules et donc on a de fortes chances pour qu'une seule transformation par cellule soit suffisante. Cependant, on va essayer de regrouper les cellules obtenues afin de réaliser le moins de transformations possible et donc d'avoir une transformation globale assez *lisse*. En effet, si on réalise une transformation sur une trop petite zone, on peut n'avoir que quelques spots, ce qui peut conduire à une transformation aberrante et donc en discontinuité avec les

Fig. 2.2 Illustration de la partition de Voronoï pour un ensemble de 10 points de $\mathbb{R}^2$.



transformations voisines.
Rappelons ici que les zones sont considérées comme homogènes si les erreurs, c'est-à-dire les distances entre points appariés (après transformation globale), sont homogènes à l'intérieur de la zone. Pour deux zones, nous allons donc prendre en compte la différence entre les moyennes de ces distances dans les deux zones. Plus cette différence est petite, plus les zones ont des chances de pouvoir être appariées par la même transformation. Cependant, afin d'obtenir une partition dans laquelle les zones sont assez *compactes*, nous allons utiliser les trois critères suivants :

- la différence des distances moyennes entre les zones,
- le nombre de sommets communs entre les zones,
- le nombre de cellules résultant de la fusion des deux zones.

On envisage uniquement la fusion des zones voisines (au moins deux sommets communs). Afin de réaliser un critère unique, nous allons tout d'abord (comme nous l'avons vu précédemment) ramener chaque critère dans l'intervalle $[0, 1]$.
Pour deux zones i et j dont les distances moyennes sont respectivement μ_i et μ_j, le critère normalisé entre 0 et 1 est le suivant :

$$crit_d(i,j) = \frac{\mu_i - \mu_j}{min_d - max_d} - \frac{max_d}{min_d - max_d},$$

$$\text{où} \quad min_d = \min_{k \neq l}(\mu_k - \mu_l) \quad \text{et} \quad max_d = \max_{k \neq l}(\mu_k - \mu_l).$$

Pour deux zones i et j dont le nombre de sommets communs est ns_{ij}, le critère normalisé est le suivant :

$$crit_s(i,j) = \frac{ns_{ij}}{max_{ns} - min_{ns}} - \frac{min_{ns}}{max_{ns} - min_{ns}},$$

$$\text{où} \quad min_{ns} = \min_{k \neq l}(ns_{kl}) \quad \text{et} \quad max_{ns} = \max_{k \neq l}(ns_{kl}).$$

Enfin, pour deux zones i et j dont la fusion engendrerait une zone contenant nc_{ij} cellules, le critère normalisé est le suivant :

$$crit_c(i,j) = \frac{nc_{ij}}{min_{nc} - max_{nc}} - \frac{max_{nc}}{min_{nc} - max_{nc}},$$

$$\text{où} \quad min_{nc} = \min_{k \neq l}(nc_{kl}) \quad \text{et} \quad max_{nc} = \max_{k \neq l}(nc_{kl}).$$

Le critère global peut alors être exprimé sous la forme suivante :

$$crit(i,j) = \alpha crit_d(i,j) + \beta crit_s(i,j) + \gamma crit_c(i,j),$$

où (α, β, γ) sont des réels qui réalisent la pondération entre les différents termes. On va donc chercher à trouver un critère minium, c'est-à-dire pour lequel la différence entre les distances moyennes est minimum, le nombre de sommets communs est maximum et le nombre de cellules fusionnées est minimum. Le but est d'obtenir des zones après regroupement homogènes et compactes. On calcule ce critère pour tous les couples de zones voisines et on regroupe celles pour lesquelles $crit(i,j)$ est minimum.
Afin de garder des zones de taille raisonnable, on arrête l'algorithme quand le prochain regroupement amènerait à regrouper plus du quart des cellules de départ. En effet, les landmarks sont régulièrement répartis donc rassembler plus du quart des cellules de départ conduirait à chercher à trouver une transformation qui soit applicable dans plus du quart de la surface du gel. Or, d'après les observations réalisées, il semble qu'il serait impossible de trouver une transformation qui soit précise pour une zone qui représenterait plus du quart du gel.

2.2.2.2.3 Optimisation des transformations locales Les zones obtenues par l'algorithme de regroupement précédemment défini sont ensuite alignées indépendamment par l'algorithme décrit au paragraphe 2.2.2. On obtient ainsi autant de couples de paramètres $(\mathbf{T}_g^i, \mathbf{t}_g^i)$ que de zones obtenues après regroupement.

2.3 Extension à plus de deux gels : généralisation de Procuste généralisé

Nous avons vu au paragraphe 2.1 que nous voulions être capables d'aligner plus de deux gels mais sans avoir à définir un gel de référence auquel seraient alignés tous les autres gels. Nous allons donc généraliser la méthode décrite dans la partie précédente de sorte à ce que chaque gel soit apparié, le mieux possible, avec l'ensemble de tous les autres gels. Pour cela, nous allons nous baser sur une généralisation de Procuste. Dans cette partie, on considérera toujours que le paramètre de dilatation est inutile.

2.3.1 Analyse procrustéenne généralisée

On veut généraliser l'analyse procrustéenne à K tableaux ($K > 2$) de dimension $n \times p$, $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_K)$. On doit donc optimiser :

- K matrices de rotation/réflexion, $\mathbf{T}^i$, pour $i \in \{1, \ldots, K\}$,
- K vecteurs colonne de translation, $\mathbf{t}^i$, pour $i \in \{1, \ldots, K\}$.

Le critère à minimiser devient alors :

$$L_G = \sum_{k<l}^{K} tr\left[(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_l)'(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_l)\right], \tag{2.18}$$

où $\hat{\mathbf{X}}_k = \mathbf{X}_k\mathbf{T}^k + \mathbf{1}_n\mathbf{t}^k$.
Malheureusement, il n'existe pas de solution analytique directe, il faut donc passer par un algorithme itératif. Pour cela, il existe plusieurs méthodes (Kristof & Wingersky, 1971; Gower, 1975; Ten Berge, 1977; Commandeur, 1991; Dijksterhuis & Gower, 1991). Nous allons décrire celle que nous allons utiliser et qui est plus précisément étudiée dans Commandeur (1991). Ce choix a été fait car c'est la méthode qui s'insère le plus naturellement dans l'AG. Cette méthode repose sur une modification du critère de l'équation (2.18) :

$$\begin{aligned}
L_G &= \sum_{k<l}^{K} tr\left[(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_l)'(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_l)\right] \\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{l=1}^{K} tr\left[(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_l)'(\hat{\mathbf{X}}_k - \hat{\mathbf{X}}_l)\right] \\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{l=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] + \frac{1}{2}\sum_{k=1}^{K}\sum_{l=1}^{K} tr[\hat{\mathbf{X}}_l'\hat{\mathbf{X}}_l] - \sum_{k=1}^{K}\sum_{l=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_l] \\
&= K\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] - \sum_{k=1}^{K}\sum_{l=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_l] \\
&= K\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] - K\sum_{k=1}^{K} tr\hat{\mathbf{X}}_k'(K^{-1}\sum_{l=1}^{K} tr\hat{\mathbf{X}}_l')
\end{aligned}$$

$$\text{Si on pose} \quad \mathbf{Z} = \frac{1}{K}\sum_{k=1}^{K}\hat{\mathbf{X}}_k \quad \text{alors}$$

$$\begin{aligned}
L_G &= K\left(\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] - \sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}]\right) \\
&= K\left(\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] + \sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}] - 2\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}]\right) \\
&= K\left(\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] + K(K^{-1}\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}]) - 2\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}]\right) \\
&= K\left(\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] + Ktr[\mathbf{Z}'\mathbf{Z}] - 2\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}]\right) \\
&= K\left(\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\hat{\mathbf{X}}_k] + \sum_{k=1}^{K} tr[\mathbf{Z}'\mathbf{Z}] - 2\sum_{k=1}^{K} tr[\hat{\mathbf{X}}_k'\mathbf{Z}]\right) \\
&= K\sum_{k=1}^{K} tr\left[(\hat{\mathbf{X}}_k - \mathbf{Z})'(\hat{\mathbf{X}}_k - \mathbf{Z})\right].
\end{aligned}$$

Le critère (2.18) devient donc :

$$L_G = K \sum_{k=1}^{K} tr\left[(\hat{\mathbf{X}}_k - \mathbf{Z})'(\hat{\mathbf{X}}_k - \mathbf{Z})\right], \tag{2.19}$$

où $\mathbf{Z}$ représente le tableau moyen des $\hat{\mathbf{X}}_k$. D'un point de vue géométrique, les individus de $\mathbf{Z}$ sont les centres de gravité de tous les points appariés correspondants.
Pour réaliser la minimisation, à chaque étape, on optimise un des $\hat{\mathbf{X}}_k$ en gardant les autres fixés. Les propriétés de convergence de cet algorithme ont été démontrées dans Commandeur (1991).

2.3.2 Application aux gels d'électrophorèse 2D et insertion dans l'AG

Nous allons maintenant utiliser Procuste généralisée pour apparier plus de deux gels d'électrophorèse 2D. Les $\mathbf{X}_k$ représenteront alors les différents gels. On aura toujours $p = 2$. Quant au nombre d'individus (c'est-à-dire de spots), il est évidemment différent entre les gels, on notera n_k le nombre de spots pour le gel $\mathbf{X}_k$. Bien que l'on ait maintenant à notre disposition l'analyse Procrustéenne généralisée, il reste une grosse difficulté, la gestion des appariements. En effet, il n'y a pas le même nombre de spots par gel, il n'y a pas le même nombre de spots appariés par gel, certains spots peuvent être appariés dans certains gels mais pas dans les autres, certains spots peuvent n'être appariés à aucun autre spot, etc...
Cela pose donc une difficulté, notamment pour la construction du gel moyen $\mathbf{Z}$. Etant donné que le but de notre méthode est précisément de permettre, pour tous les gels, tous les types d'appariement possibles (appariement à aucun gel, à tous les gels, à certains gels), à chaque étape, on doit considérer tous ces types d'appariement.
Pour gérer cela, nous allons construire une matrice $\mathbf{A}$ qui a autant de lignes qu'il y a d'appariements, y compris ceux que l'on nommera *appariements nuls* qui correspondent à des spots appariés à aucun autre. Pour chaque appariement, on trouve les numéros (arbitrairement attribués au début de l'algorithme) des spots appariés pour les gels concernés et des zéros pour les gels non concernés par cet appariement.
Par exemple, si on a trois gels et qu'on décide d'apparier le quatrième spot du gel 1 avec le troisième spot du gel 3 et avec aucun spot du gel 2, la ligne correspondant à cet appariement sera (4,0,3). La première colonne contient les numéros des spots du premier gel, la deuxième colonne les numéros des spots du deuxième gel, etc...
Au début de l'algorithme, il n'y a que des appariements nuls, la matrice $\mathbf{A}$ compte alors K colonnes (une par gel) et $\sum_{k=1}^{K} n_k$ lignes (une ligne par spot). On peut alors la représenter sous la forme décrite dans la Fig. 2.3.

Si on décide d'apparier deux spots, il suffit de fusionner les lignes correspondantes en ne gardant les zéros que pour les colonnes où il y avait un zéro dans les deux lignes fusionnées. Alors, le gel moyen, $\mathbf{Z}$, sera construit en utilisant $\mathbf{A}$ et en faisant la moyenne par ligne, sans tenir compte des zéros. Si on revient à l'exemple précédent, on avait la ligne (4,0,3) ; pour calculer les coordonnées du spot correspondant à cet appariement dans $\mathbf{Z}$, on calculera les coordonnées du centre de gravité du quatrième spot du gel 1 et du troisième spot du gel 2. La matrice $\mathbf{Z}$ aura donc autant de lignes que $\mathbf{A}$ et deux colonnes (une par coordonnée).
On gère parallèlement la matrice $\mathbf{Z}^l$ qui contient les coordonnées des centres de gravité des landmarks. On note $\mathbf{X}_k^l$ la matrice contenant les coordonnées des landmarks du gel $\mathbf{X}_k$. La

Fig. 2.3 Représentation de la matrice **A** des appariements de K gels au début de l'algorithme.

$$\begin{vmatrix} 1 & 0 & . & 0 & . & 0 \\ 2 & 0 & . & 0 & . & 0 \\ . & . & . & 0 & . & . \\ n_1 & . & . & 0 & . & 0 \\ 0 & 1 & . & 0 & . & 0 \\ 0 & 2 & . & 0 & . & 0 \\ . & . & . & . & . & . \\ 0 & n_2 & . & 0 & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & . & 1 & . & 0 \\ 0 & 0 & . & 2 & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & . & n_k & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & . & 0 & . & 1 \\ 0 & 0 & . & 0 & . & 2 \\ . & . & . & . & . & . \\ 0 & 0 & . & 0 & . & n_K \end{vmatrix}$$

matrice $\mathbf{Z}^l$ va nous servir (comme précédemment) à initialiser l'algorithme.
On peut alors décrire l'algorithme d'alignement par le pseudo-code de la Fig. 2.4.

Le critère d'arrêt peut être un nombre d'itérations mais plus judicieusement, on peut arrêter l'algorithme quand il n'y a plus d'évolution significative des paramètres de transformation.
Comme pour le cas de deux gels, l'alignement se fait en deux étapes : on réalise tout d'abord l'alignement global puis on utilise les erreurs obtenues pour réaliser une partition des gels afin d'optimiser des transformations locales. Pour ces deux étapes, on peut utiliser l'algorithme décrit dans la Fig.2.4. Cependant, pour accélérer l'alignement, on peut se contenter, pour l'étape globale, de réaliser une analyse Procuste généralisée sur les landmarks et d'utiliser les paramètres obtenus pour la transformation globale. Puisque les landmarks sont supposées être bien choisis, la transformation obtenue doit être d'une qualité tout à fait suffisante pour définir les zones homogènes du gel. Ensuite, on applique l'algorithme complet sur chacune des zones isolées.

2.4 Une première approche de la convergence : l'observation

Maintenant que les étapes de l'algorithme d'alignement de gels sont établies, nous allons voir ce qui se passe concrètement sur un jeu de données réelles. Cette étape va nous permettre d'étudier l'efficacité de la méthode mise en place. De plus, nous pourrons commencer à avoir un premier regard sur la convergence de tels algorithmes par l'observation de l'évolution des paramètres et par l'étude de la population finale.

Fig. 2.4 Pseudo-code de l'AG permettant l'alignement de plus de deux gels.

Initialisation
pour $k = 1, 2, \ldots, K$ **faire**
    $(\mathbf{T}^k, \mathbf{t}^k)$=Procuste$(\mathbf{X}_k^l, \mathbf{Z}^l)$
fin
Alignement des gels
Tant que critère de fin non obtenu **faire**
    pour $k = 1, 2, \ldots, K$ **faire**
        On initialise la population avec les paramètres $(\mathbf{T}^k, \mathbf{t}^k)$ courants.
        On applique l'AG décrit à la Fig. 2.1 aux matrices $\hat{\mathbf{X}}_k$ et $\mathbf{Z}$.
        On obtient de nouveaux paramètres $(\mathbf{T}^k, \mathbf{t}^k)$ et des listes d'appariements entre $\hat{\mathbf{X}}_k$ et $\mathbf{Z}$.
        $\hat{\mathbf{X}}_k \leftarrow \hat{\mathbf{X}}_k\mathbf{T}^k + \mathbf{1}_n\mathbf{t}^k$
        On met à jour $\mathbf{Z}$.
    fin
fin

2.4.1 Les données

Dans cette partie, nous allons étudier un jeu de trois gels afin d'y appliquer la méthode généralisée tout en gardant des résultats simples à suivre, notamment d'un point de vue graphique. Les trois gels que l'on considère sont le résultat d'une étude portant sur des cellules-souches adultes du tissu adipeux. Cette étude a été menée dans l'équipe CHU-CNRS du Professeur Sylvain Lehman dans le cadre du projet européen GENOSTEM. Il s'agit de cellules en culture que l'on réalise pour trois individus différents. L'objectif est de comparer les différences entre individus sans différence particulière de conditions.
Pour cela, on prélève 20 μg de protéines (extraites des cellules totales). On les fait alors migrer dans une première dimension sur des bandelettes contenant un gradient linéaire de pH allant de 3 à 10 d'après les fabricants. En réalité, le pouvoir tampon des protéines a tendance à réduire le gradient (plutôt entre 4 et 9) et à modifier sa linéarité. Lors de cette migration, les protéines s'immobilisent à leur point isoélectrique (c'est-à-dire à l'endroit où la charge électrique du milieu compense leur charge propre). Cette dimension correspond à l'axe horizontal des gels que nous allons présenter, les pH acides se trouvant du côté gauche et les pH basiques du côté droit. La deuxième migration permet de séparer les protéines en fonction de leur poids moléculaire apparent. On parle de poids moléculaire apparent (contrairement aux spectromètres de masse où on obtient le poids moléculaire réel) car de nombreux facteurs interviennent et influencent la migration, par exemple la phosphorylation d'une protéine provoque une migration plus rapide. Cette migration correspond à l'axe vertical de nos représentations avec les hauts poids moléculaires en haut des graphiques.
Une fois les deux migrations réalisées, on réalise une coloration au nitrate d'argent. Cette coloration a été choisie en raison de sa grande sensibilité. Les gels sont alors numérisés sur un scanner classique à transmission. La détection des spots est alors réalisée grâce au logiciel ImageMaster 2D Platinum v5.0 (Amersham Biosciences, Uppsala, Suède). Les résultats de cette détection sont présentés dans la Fig. 2.5. On a, au total, 994 spots dans le premier gel, 915 dans le deuxième et 1022 dans le dernier.

Fig. 2.5 Représentation de la superposition directe des trois gels de l'expérience. Chaque point correspond à la détection d'un spot. Le premier gel est représenté en croix bleues, le deuxième en cercles verts et le troisième en croix rouges. L'unité des deux axes est le pixel.

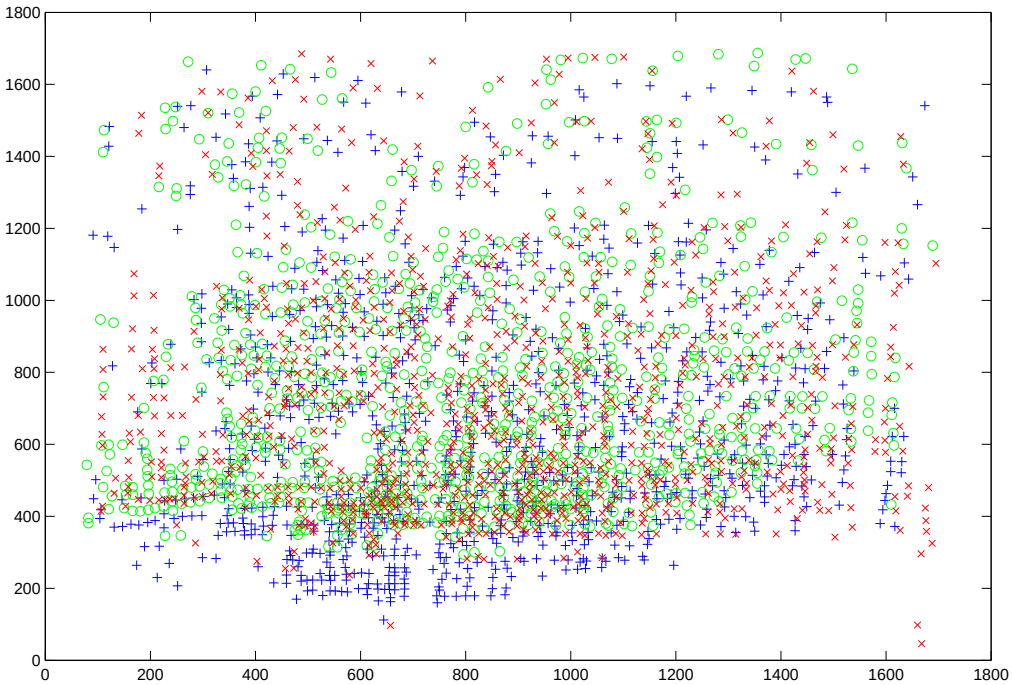
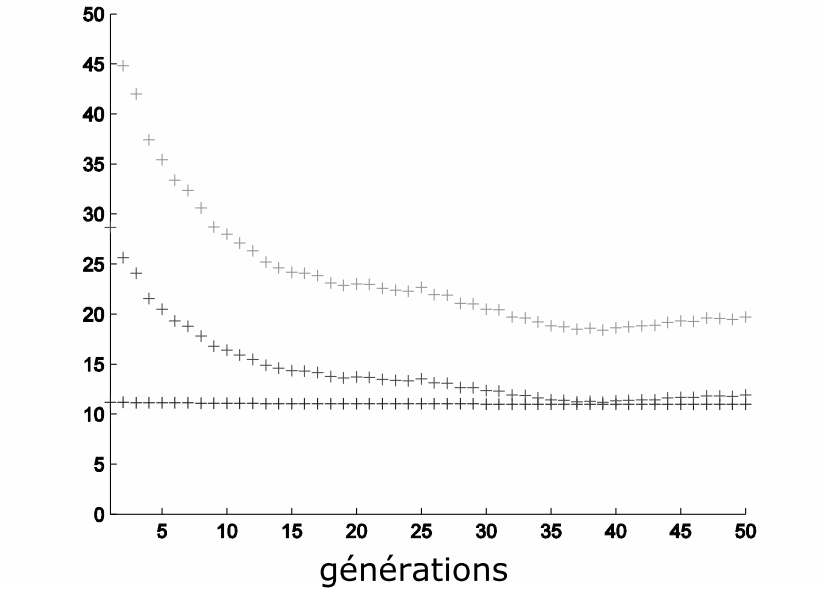


Fig. 2.6 Evolution des différents termes de la fitness pour l'alignement de deux gels d'électrophorèse 2D. Chaque point représente la moyenne des individus de la génération courante. La courbe du haut représente l'évolution de l'erreur moyenne, la courbe du bas, celle du nombre d'appariements et la courbe intermédiaire, celle du critère global. L'unité de l'axe des ordonnées est arbitraire.



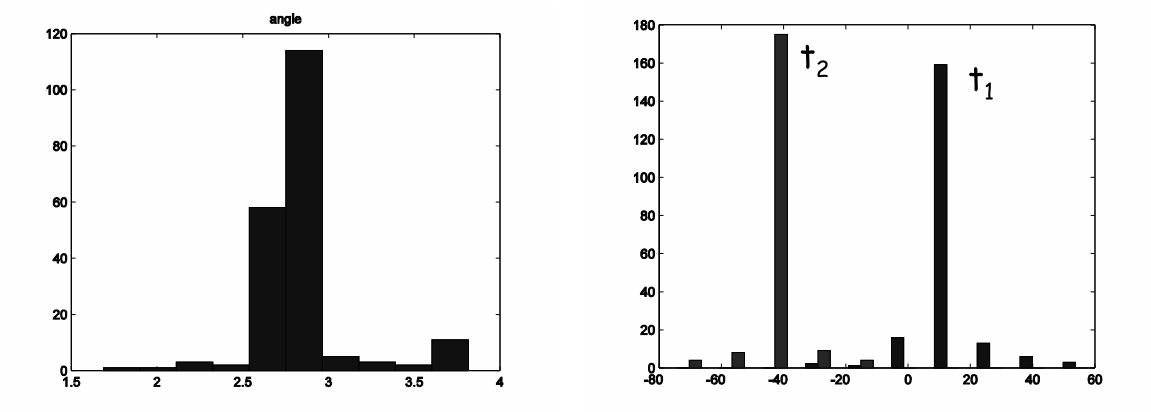
2.4.2 Résultats de l'alignement global

2.4.2.1 Cas de deux gels

Nous allons, dans un premier temps, nous focaliser sur l'alignement de deux des gels. En effet, nous avons vu que pour aligner plus de deux gels, nous réalisons des itérations jusqu'à ce que la transformation n'évolue plus, l'étude des populations finales n'est donc pas intéressante du point de vue des paramètres de transformation. En revanche, quand on aligne deux gels, on ne réalise qu'une seule transformation dont nous pouvons étudier les composantes finales.
Voyons dans un premier temps l'évolution des différents termes du critère au cours des générations (voir Fig. 2.6). On constate que le nombre d'appariements est très stable. En fait, il est très élevé dès le début et ne diminue que très peu. Cela est dû essentiellement au fait qu'on impose, pour chaque transformation d'apparier chaque spot d'un gel à un spot de l'autre gel avant d'éliminer seulement les appariements pour lesquels la distance est supérieure à un seuil. En revanche, pour la distance entre spots elle est très élevée au départ, puis elle diminue rapidement et finit par se stabiliser.

Etudions maintenant les paramètres de la population finale de l'AG présentés dans la Fig. 2.7. On constate que ces distributions sont très concentrées autour de leur mode. En effet, pour l'angle de rotation, 90% des valeurs se situent entre 2.5° et 3°, ce qui est à peine discernable du point de vue du résultat de la transformation. Pour la translation horizontale, 88% des solutions portent la valeur -39 pixels et pour la translation verticale, la valeur 12 pixels est prise par 79% des individus de la population finale. Si on considère, en outre, que

Fig. 2.7 Distribution des paramètres des transformations pour les 200 solutions de la population finale. La distribution des angles de rotation (en degrés) est représentée dans la graphique de gauche et la distribution des translations (en pixels) dans le graphique de droite (t_1 pour la translation verticale et t_2 pour la translation horizontale).



le taux de mutation réaugmente à la fin de l'algorithme, on constate que la population finale a bien convergé autour d'une valeur pour chaque paramètre de la transformation.
Une telle information doit être examinée avec précaution. En effet, la concentration des valeurs indique clairement que la transformation obtenue correspond à un optimum mais il est impossible de savoir avec certitude si cet optimum est global ou non. Cependant, quand on ne dispose pas d'information sur la nature de l'optimum global, une telle distribution des paramètres dans la population finale est très intéressante et justifie que l'on arrête l'algorithme. Notons également que la solution modale de la population finale correspond à la meilleure solution dans cette population du point de vue de la fitness. Nous verrons dans le paragraphe 3 une utilisation plus précise de ce type de propriété.

2.4.2.2 Cas de trois gels

Revenons maintenant au cas de trois gels. La Fig. 2.5 montre bien la nécessité d'un réalignement des gels mais aussi d'une transformation locale. En effet, même *à l'œil*, on arrive à repérer des spots qui devraient être appariés mais si on recherche de tels appariements dans différentes parties des gels, on constate que l'application d'une transformation commune ne saurait permettre de réaliser les bons appariements dans tout le gel. De plus, en observant cette figure, on voit que certaines zones sont très différentes entre gels. Par exemple, dans la partie inférieure, on voit que le gel 1 (en bleu) comporte beaucoup plus de spots que les autres et cela ne résulte pas uniquement d'une translation. On identifie alors facilement des spots qui n'auront de correspondant dans aucun autre gel. Voyons donc maintenant les résultats de l'application de notre AG à ces trois gels.

2.4.2.2.1 Résultats de l'alignement global Nous allons, dans un premier temps, étudier les sorties de l'alignement global des trois gels. Nous savons que cet alignement ne saurait être utilisé seul mais il va nous servir à initialiser les alignements locaux et à définir des zones homogènes dans les gels, comme nous l'avons expliqué au paragraphe 2.3.2. On peut représenter les résultats de cet alignement par les deux graphiques de la Fig. 2.8.

Dans la partie supérieure, nous avons relié les spots appariés, notons que nous avons représenté les gels avant transformation, ce qui permet de mieux visualiser la qualité de cette transformation. En effet, on voit que certaines zones sont déjà bien appariées mais en observant globalement le graphique, on constate que les segments reliant les spots appariés peuvent avoir des directions très différentes, même dans une même zone. Ceci peut sembler étrange étant donné que l'on applique la même transformation à tout le gel. Cependant, rappelons qu'une fois la transformation réalisée, on apparie les spots les plus proches. Si ces spots étaient tous exactement superposés, après transformation, on aurait alors des segments parallèles et plus les spots appariés sont encore éloignés après la transformation plus les segments les reliant seront *désordonnés*. L'observation montre donc déjà que la transformation globale n'est pas optimale.
De plus, si on observe la partie inférieure de la Fig. 2.8, on constate que les erreurs moyennes varient entre 0 et 55 à l'intérieur des petites zones. Notons ici qu'il n'y a pas forcément des appariements dans chacune des petites cellules. Dans ce cas, on applique la moyenne des distances dans les cellules voisines, c'est pourquoi le graphique doit être étudié globalement et non dans le détail. Ceci est surtout vrai vers les bords du graphique où il y a peu de spots. En effet, si on a une forte distance dans une cellule, elle peut provoquer une impression de grande distance dans les cellules voisines. Cependant, ce graphique nous permet d'ores et déjà de repérer des zones qui semblent homogènes dans les gels, du point de vue de ces erreurs. Mais nous avons vu que pour définir précisément ces zones homogènes, nous utilisons une partition par cellules de Voronoï puis un regroupement en fonction de différents critères dont le plus important est la différence entre les erreurs moyennes dans les cellules.

2.4.2.2.2 Recherche de zones homogènes La Fig. 2.9 montre la partition réalisée et le regroupement des cellules homogènes. Dans cet exemple, 34 landmarks ont été utilisés. On peut noter que de tels landmarks peuvent être facilement retrouvés, comme nous l'avons vu précédemment, puisqu'il est tout à fait possible de repérer des appariements évidents à partir de la représentation des coordonnées des spots détectés. Le regroupement des cellules permet de retrouver les zones qui semblaient homogènes dans la Fig. 2.8. On constate que l'algorithme permet de trouver des zones de tailles variables allant d'une seule cellule à neuf cellules. Il semble donc bien s'adapter à la structure du gel. Cependant, on pourrait regretter l'apparition de deux zones ne contenant qu'une seule cellule, d'autant plus que ceci est pénalisé par l'algorithme de regroupement. Nous verrons dans la suite quelle est leur influence.

2.4.2.2.3 Résultats des alignements locaux Une fois la définition des zones homogènes réalisée, il ne reste qu'à appliquer l'AG d'alignement à chacune de ces zones. Les résultats sont présentés dans la Fig. 2.10. On constate immédiatement, dans la partie supérieure, que l'aspect global de la transformation des gels est beaucoup plus régulier, ce qui pourrait sembler paradoxal étant donné que l'on a un jeu de paramètres de transformation dans chaque zone. Mais, comme nous l'avons indiqué précédemment, une telle homogénéité reflète la grande proximité des points après transformation, donc la bonne qualité de la transformation. De plus, et c'est le plus intéressant, on constate une grande continuité de la transformation d'une zone à l'autre. Il est impossible, d'après ce graphique, de repérer quelles étaient les zones de transformation, la transition se fait *en douceur*. Cela nous laisse

Tab. 2.1 Pourcentage de spots restant isolés après alignement global et local

gel	global	local
1	23%	23%
2	13%	11%
3	22%	19%

penser que le découpage et la transformation obtenus sont de bonne qualité, malgré les deux cellules isolées. De plus, la forme ovoïde que l'on obtient pour la transformation est tout à fait conforme à ce que les biologistes observent habituellement. Notons d'ailleurs que nos appariements ont été validés par les biologistes.
Si on étudie les appariements, dans l'alignement global, on avait réalisé 980 appariements alors qu'il n'en reste que 956 après alignement local. Cependant, on a plus d'appariements des trois gels dans l'alignement local : 51% des appariements concernent les trois gels dans le cas local alors qu'il n'y en avait que 41% pour le global. On a ainsi moins de points isolés, comme le montre la Tab. 2.1.

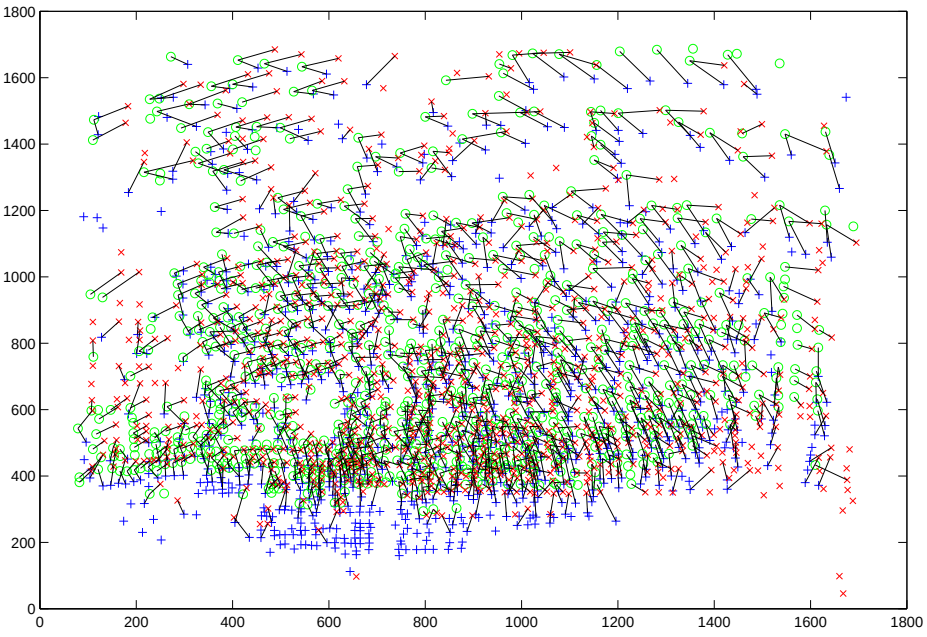
Pour ce qui est de la distance finale, si on observe les parties inférieures des Fig. 2.8 et 2.10, on constate bien une diminution globale des erreurs. Pour ce qui est de la moyenne de ces distances, elle est de 12 pixels pour l'alignement global et de 9 pixels pour l'alignement local, cette différence peut sembler minime mais nous avons déjà montré la grande amélioration provoquée par l'alignement local et cette différence ne doit donc pas être sous-estimée.

2.4.3 Conclusion

En conclusion, à supposer que cela était nécessaire, cette application montre tout d'abord l'apport d'une transformation locale par rapport à une transformation globale des gels. L'importance de cet apport est d'autant plus forte que le découpage des gels est très flexible (à condition d'avoir indiqué suffisamment de landmarks pour avoir une partition assez fine) et s'appuie sur un critère très approprié, l'erreur moyenne locale. La qualité de cette transformation apparaît essentiellement quand on observe globalement le résultat des transformations locales. Alors que l'on n'impose aucune contrainte de continuité entre les transformations de zones voisines, on observe, dans l'ensemble du gel, une transformation tout à fait régulière qui, de plus, est conforme à ce que l'on s'attend à trouver pour ce type de gels. Le découpage et le rassemblement en zones que nous avons mis au point semblent donc tout à fait efficaces.
Le deuxième apport de notre méthode est sans doute la grande souplesse concernant la structure des appariements. Alors que dans la grande majorité des méthodes existantes, un gel de référence doit être désigné (ou construit) pour pouvoir aligner plus de deux gels, dans notre méthode, tous les gels jouent le même rôle et on peut ainsi identifier des appariements entre n'importe quelle combinaison de gels.
Actuellement, la faiblesse de notre algorithme réside dans le temps de calcul. En effet, il faut environ 8h sur un PC de bureau classique pour réaliser l'alignement local. On pourrait certainement réduire notablement ce temps de calcul en programmant dans un autre langage ou en utilisant un ordinateur plus performant. Ce temps est, de toutes façons, négligeable

devant le temps nécessaire à un biologiste pour aligner les gels.
Dans les perspectives, il pourrait être intéressant de concevoir une interface graphique qui permettrait au biologiste de visualiser rapidement les résultats de l'alignement.

Fig. 2.8 Résultats de l'alignement global des trois gels. Le graphique supérieur représente les appariements réalisés sur les trois gels. Les codes couleurs sont les mêmes que pour la Fig. 2.5 et les spots appariés sont reliés par des segments. Pour la partie inférieure, chaque petite case du graphique représente une petite zone des gels et la couleur dépend de la distance moyenne (ou erreur moyenne) pour les appariements se situant dans cette zone.



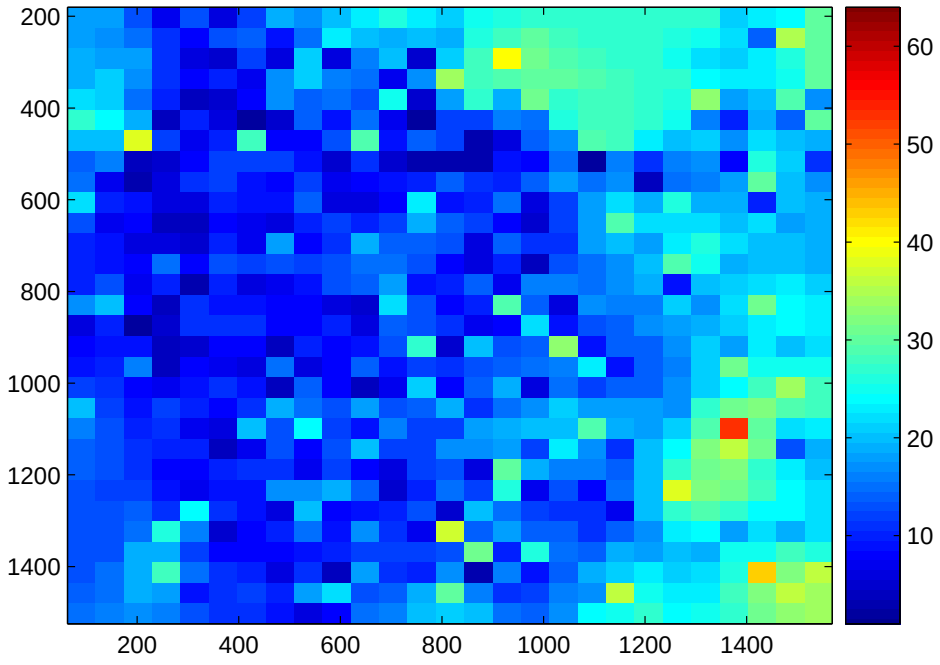


Fig. 2.9 Partitionnement de la surface des gels par utilisation des cellules de Voronoï et regroupement des cellules (une couleur par groupe) par application de l'algorithme décrit au paragraphe 2.2.2.2.

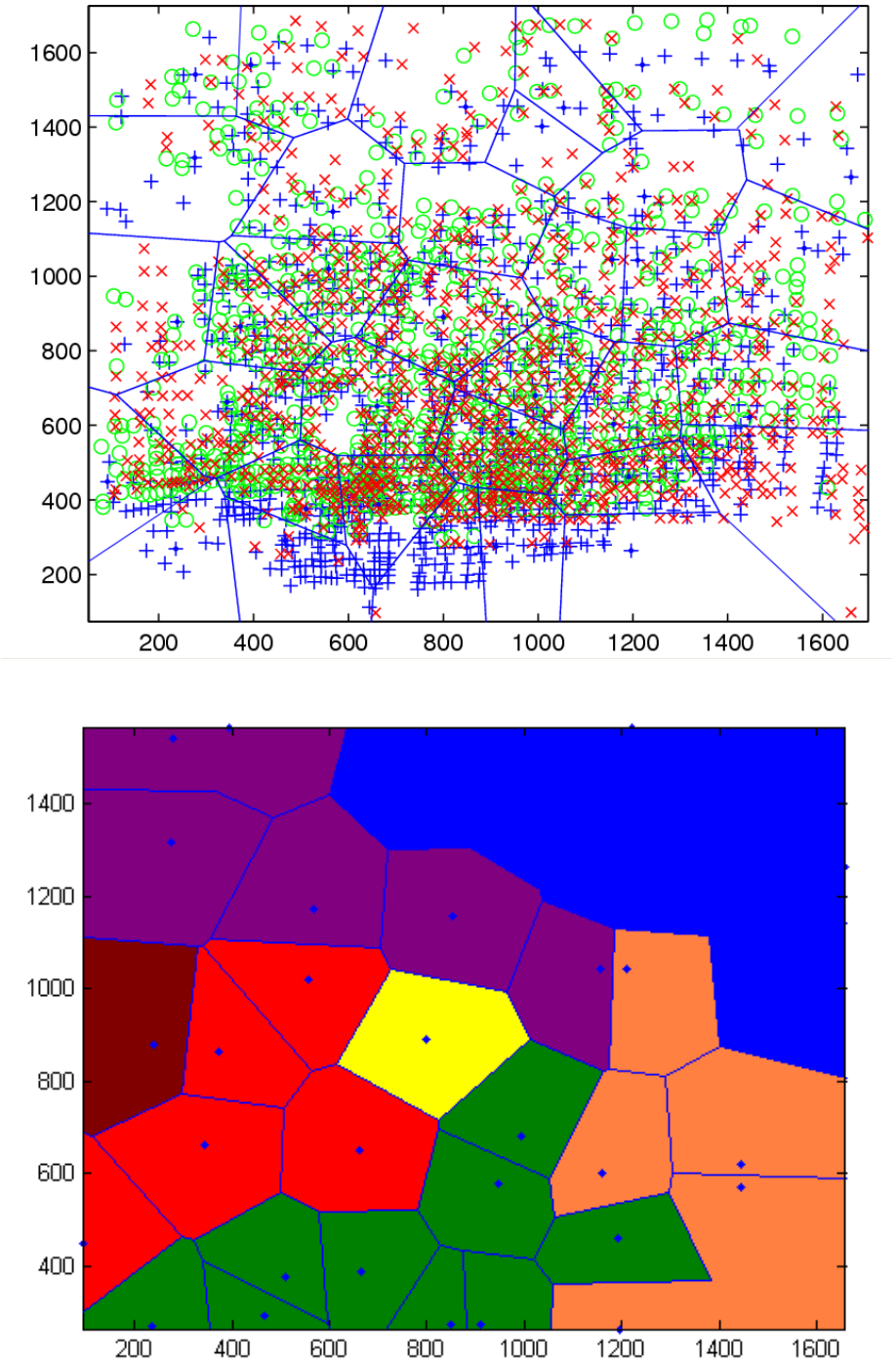
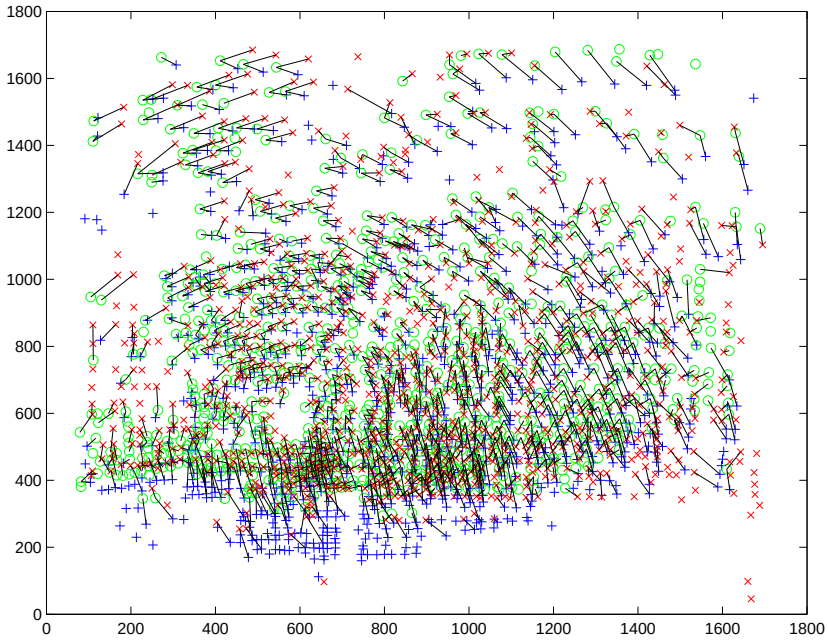
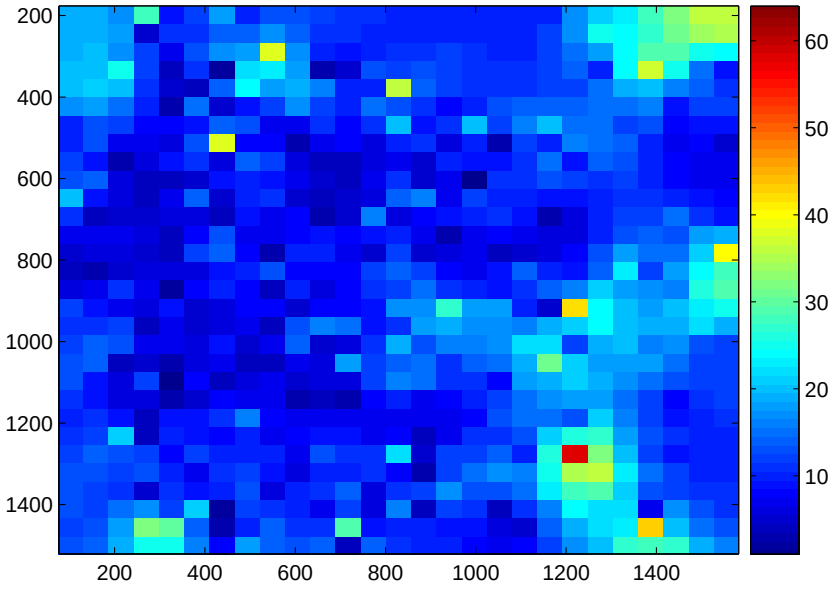


Fig. 2.10 Résultats de l'alignement local des zones homogènes pour les trois gels. Le graphique supérieur représente les appariements réalisés sur les trois gels. Les codes couleurs sont les mêmes que pour la Fig. 2.5 et les spots appariés sont reliés par des segments. Pour la partie inférieure, chaque petite case du graphique représente une petite zone des gels et la couleur dépend de la distance moyenne (ou erreur moyenne) pour les appariements se situant dans cette zone.





Troisième partie

Convergence des AG : de la théorie à la mise en application

Chapitre 1

Démonstration de la convergence théorique : généralisation au cas non homogène et aux mutations *dirigées*

Comme nous l'avons annoncé dans le paragraphe 2.2.2, nous allons maintenant étendre les résultats de Bhandari *et al.* (1996) au cas d'une chaîne de Markov non homogène (taux de mutation variable) et aux mutations dirigées. Pour cela, nous allons nous baser sur le premier exemple d'application présenté qui concerne les données de SELDI.

1.1 Codage des solutions

Dans un premier temps, nous allons rappeler comment est effectué le codage des solutions. Dans cette application, un chromosome représente un comité, c'est-à-dire un ensemble de pics avec leurs seuils associés. Pour pouvoir appliquer les opérateurs (sélection, croisement, mutation) simplement, on fait en sorte que la longueur des chromosomes soit constante. Pour ce faire, on doit se donner N_m, nombre maximum de pics dans le comité.
Ainsi, pour chaque membre du comité, il faut donner le numéro du pic et le seuil associé. On obtient donc des chromosomes ayant une longueur de $2N_m$. Si, un comité contient moins de N_m pics, on utilise la valeur NA pour remplir les éléments vides du vecteur.
Rappelons maintenant les *alphabets* dans lesquels sont écrits chaque caractère du chromosome, c'est-à-dire les ensembles de définition des éléments du vecteur. On note p le nombre total de pics parmi lesquels on peut choisir les membres du comité et n le nombre de spectres. On a alors besoin de $p + 2$ alphabets, en effet :

$$\mathcal{A}_p = \{1, 2, \ldots, p\}$$
$$\mathcal{A}_{t_1} = \{t_{11}, t_{12}, \ldots, t_{1c_1}\} \text{ où } c_1 \leq n$$
$$\ldots$$
$$\mathcal{A}_{t_i} = \{t_{i1}, t_{i2}, \ldots, t_{ic_i}\} \text{ où } c_i \leq n$$
$$\ldots$$
$$\mathcal{A}_{t_p} = \{t_{p1}, t_{p2}, \ldots, t_{pc_p}\} \text{ où } c_p \leq n$$
$$\mathcal{A}_N = \{\text{NA}\}$$

où t_{ij} désigne le $j^{\text{ème}}$ seuil possible pour le $i^{\text{ème}}$ pic et c_i désigne le nombre de seuils possibles pour le $i^{\text{ème}}$ pic.
Notons $\mathcal{S}$, l'ensemble des chromosomes de cette forme. Alors, un chromosome, $S \in \mathcal{S}$, pour un comité comportant K pics ($K \leq N_m$) contiendra donc $2(N_m - K)$ valeurs NA et s'écrira de la manière suivante :

$$S = \{p_1, t_{p_1 j_1}, p_2, t_{p_2 j_2}, \ldots, p_K, t_{p_K j_K}, \text{NA}, \text{NA}, \ldots, \text{NA}, \text{NA}\}$$

où $p_i \in \mathcal{A}_p$ et $t_{p_i j} \in \mathcal{A}_{t_{p_i}}$ avec $(i = 1, 2, \ldots, K)$ et $(j \in \{1, 2, \ldots, c_{p_i}\})$.

1.2 Dénombrement des solutions possibles

Afin de rendre possible ces dénombrements, nous allons considérer $c_i = n, \forall i \in \{1, 2, \ldots, p\}$. On sait que $c_i \leq n$, $\forall\, i \in \{1, 2, \ldots, p\}$, on obtiendra donc des majorants des nombres de possibilités.

Nombre total de chromosomes possibles avec répétition des pics autorisée
Pour un chromosome comportant K pics, pour le choix de chaque pic, on a p possibilités et pour le choix de chaque seuil, on a n possibilités. Puisqu'on autorise les répétitions, le choix de chaque pic (avec son seuil) est indépendant du choix des autres pics, on a donc $(pn)^K$ chromosomes de K pics possibles.
Le nombre total de chromosomes possibles est :

$$\text{card}(\mathcal{S}) = \mathcal{N}_1 = \sum_{K=1}^{N_m} (pn)^K = \frac{pn(1 - (pn)^{N_m})}{1 - pn}.$$

Notons que dans la population initiale, on applique forcément le seuil optimal, donc il n'y a pas de choix pour les seuils une fois que les pics sont choisis, on a donc $\mathcal{N}_1^0 = \sum_{K=1}^{N_m} p^K$.

Nombre total de chromosomes possibles sans répétition des pics
Pour un chromosome comportant K pics, pour le choix des pics on a $\binom{p}{K}$ possibilités de choisir K pics parmi les p possibles (les pics étant utilisés en parallèle, leur ordre n'a pas d'importance). Pour chaque seuil, on a toujours c choix possibles, il y a donc $\binom{p}{K} n^K$ chromosomes possibles pour des comités de K pics.
Le nombre total de chromosomes possibles est alors :

$$\text{card}(\mathcal{S}) = \mathcal{N}_2 = \sum_{K=1}^{N_m} \binom{p}{K} n^K.$$

Pour la population initiale, $\mathcal{N}_2^0 = \sum_{K=1}^{N_m} \binom{p}{K}$.

Nombre total de populations possibles
On note $\mathcal{Q}$, l'ensemble des populations possibles. On considère $\mathcal{N} \in \{\mathcal{N}_1, \mathcal{N}_1^0, \mathcal{N}_2, \mathcal{N}_2^0\}$, alors le nombre total de populations de taille M possibles est le nombre de façons de choisir M chromosomes parmi les $\mathcal{N}$ possibles avec répétition. Or, on sait que le nombre de combinaisons avec répétition de p éléments de E avec $\text{card}(E) = n$ est $\binom{n+p-1}{p}$. Donc le nombre total

Tab. 1.1 Ordres de grandeurs des nombres de chromosomes et populations possibles pour $n = 37$, $p = 93$ et $N_m = 10$

	card($\mathcal{S}$)	card($\mathcal{Q}$)
avec répétition	2.10^{35}	2.10^{6685}
sans répétition	4.10^{28}	3.10^{5345}

de populations possibles est ici de $\text{card}(\mathcal{Q}) = \binom{M+\mathcal{N}-1}{M}$.
Pour avoir quelques ordres de grandeur, nous pouvons calculer ces différentes quantités pour un petit exemple. On considère un petit échantillon comportant $n = 37$ spectres dont on a étudié une partie contenant $p = 93$ pics avec $N_m = 10$. Alors on obtient les quantités données dans la table 1.2. Ces chiffres nous montrent essentiellement qu'il est impossible d'envisager de tester l'ensemble des solutions possibles.

Partitionnement des chromosomes
Chaque chromosome $S \in \mathcal{S}$ a une valeur de fitness $fit(S)$. On peut alors noter

$$\mathcal{C} = \{fit(S) \ : \ S \in \mathcal{S}\}.$$

On note $s = \text{card}(\mathcal{C}) \leq \mathcal{N}$. On peut alors ordonner les éléments de $\mathcal{C}$ de la manière suivante :

$$\mathcal{C} = \{F_1, F_2, \ldots, F_s\}, \text{ où } F_1 > F_2 > \ldots > F_s.$$

On peut alors réaliser une partition de $\mathcal{S} = \{\mathcal{S}_i\}$ où

$$\mathcal{S}_i = \{S \ : \ S \in \mathcal{S} \ et \ fit(S) = F_i\}, \text{ avec } i = 1, 2, \ldots, s. \tag{1.1}$$

Soit $u_i = \text{card}(\mathcal{S}_i)$ alors $\sum_{i=1}^{s} u_i = \mathcal{N}$. On a les propriétés triviales suivantes :

Propriété 1.1

$$\mathcal{S}_i \neq \emptyset, \ \forall \, i \in \{1, 2, \ldots, s\},$$

Propriété 1.2

$$\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \ pour \, i \neq j, \ \{i, j\} \in \{1, 2, \ldots, s\}^2,$$

Propriété 1.3

$$\bigcup_{i=1}^{s} \mathcal{S}_i = \mathcal{S}.$$

D'après les propriétés 1.1, 1.2 et 1.3, $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_s\}$ est bien une partition de $\mathcal{S}$. De plus, d'après la définition des $\{\mathcal{S}_i\}$ $(i = 1, \ldots, s)$ si $S_1 \in \mathcal{S}_i$ et $S_2 \in \mathcal{S}_j$ $((i, j) \in \{1, 2, \ldots, s\}^2)$ alors

$$i < j \Rightarrow fit(S_1) > fit(S_2),$$

$$i = j \Rightarrow fit(S_1) = fit(S_2),$$

$$i > j \Rightarrow fit(S_1) < fit(S_2).$$

On a, en particulier, la propriété remarquable suivante :

Propriété 1.4 *L'ensemble $\mathcal{S}_1$ contient tous les chromosomes optimaux au sens de la fitness.*

Partitionnement des populations

Soit $Q \in \mathcal{Q}$ une population quelconque contenant M chromosomes de $\mathcal{S}$. Notons qu'un même chromosome peut apparaître plusieurs fois dans une population.

Définition 1.1 *: Egalité de deux populations*
On considère deux populations comme égales si elles contiennent les mêmes chromosomes en même nombre de copies :

$$Q_1 = Q_2 \Leftrightarrow (\{S_j \in Q_1 \, et \, \sigma_{1j} = k\} \Leftrightarrow \{S_j \in Q_2 \, et \, \sigma_{2j} = k\})$$

où σ_{ij} est le nombre d'occurrences du chromosome S_j dans la population Q_i.

Définition 1.2 *: Fitness d'une population*

$$fit(Q) = \max_{S \in Q} fit(S).$$

On peut alors réaliser une partition des populations, $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_s\}$ définie de la même manière que pour les chromosomes (voir l'équation 1.1) :

$$\mathcal{Q}_i = \{Q \, : \, Q \in \mathcal{Q} \text{ et } fit(Q) = F_i\}, \; i = 1, 2, \ldots, s.$$

On a, à nouveau, les propriétés suivantes :

Propriété 1.5

$$\mathcal{Q}_i \neq \emptyset, \; \forall \, i \in \{1, 2, \ldots, s\},$$

Propriété 1.6

$$\mathcal{Q}_i \cap \mathcal{Q}_j = \emptyset \; pour \, i \neq j, \; \{i, j\} \in \{1, 2, \ldots, s\}^2$$

Propriété 1.7

$$\bigcup_{i=1}^{s} \mathcal{Q}_i = \mathcal{Q}.$$

D'après les propriétés 1.5, 1.6 et 1.7, $\{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_s\}$ est bien une partition de $\mathcal{Q}$. De plus, on a :

Propriété 1.8

$$S \in Q \; et \, Q \in \mathcal{Q}_i \Rightarrow S \in \bigcup_{j=i}^{s} \mathcal{S}_j, \; \forall i \in \{1, 2, \ldots, s\}$$

c'est-à-dire qu'un chromosome d'une population de $\mathcal{Q}_i$ ne peut avoir une fitness supérieure à F_i.

On notera dans la suite, $e_i = \text{card}(\mathcal{Q}_i)$ et Q_{ij}, $i \in \{1, 2, \ldots, s\}$ et $j \in \{1, 2, \ldots, e_i\}$, la $j^{\text{ème}}$ population de $\mathcal{Q}_i$ (les populations sont ordonnées de façon arbitraire à l'intérieur de $\mathcal{Q}_i$).

1.3 Modélisation de l'AG comme une chaîne de Markov

1.3.1 Rappel sur les Chaînes de Markov

Définition 1.3 *: Processus stochastique*
Soit $(\Omega, \mathcal{F}, P)$ un espace de probabilité, $\mathcal{T}$ un ensemble quelconque et $(\mathcal{G}, \mathcal{E})$ un espace mesurable. On appelle processus stochastique défini sur Ω, avec $\mathcal{T}$ ensemble des temps et $\mathcal{G}$ espace d'états toute famille $\{G_t\}_{t\in\mathcal{T}}$ de variables aléatoires à valeurs dans $\mathcal{G}$. La variable aléatoire G_t est appelée état à l'instant t.

Définition 1.4 *: Chaîne de Markov*
Un processus stochastique $\{G_t\}_{t\in\mathcal{T}}$ défini sur un espace de probabilité $(\Omega, \mathcal{F}, P)$ avec $\mathcal{G}$, ensemble fini ou dénombrable, espace d'états du système et $\mathcal{T}$, ensemble des paramètres du temps (ici $\mathcal{T} = \mathbb{N}$), est appelé Chaîne de Markov d'ordre r à temps $\in \mathcal{T}$ et à valeurs dans $\mathcal{G}$ si

$$P(G_t = g_t | G_{t-1} = g_{t-1}, \ldots, G_0 = g_0) = P(G_t = g_t | G_{t-1} = g_{t-1}, \ldots, G_{t-r} = g_{t-r}).$$

En d'autres termes, l'état à l'instant t ne dépend que des r états précédents.

La loi d'une chaîne de Markov d'ordre r est entièrement déterminée par sa *loi de probabilité initiale*, $\pi_0(g_i) = P(G_0 = g_i)$, $(g_i \in \mathcal{G})$ et ses *probabilités de transition* :

$$P(G_t = g_t | G_{t-1} = g_{t-1}, \ldots, G_{t-r} = g_{t-r}),$$

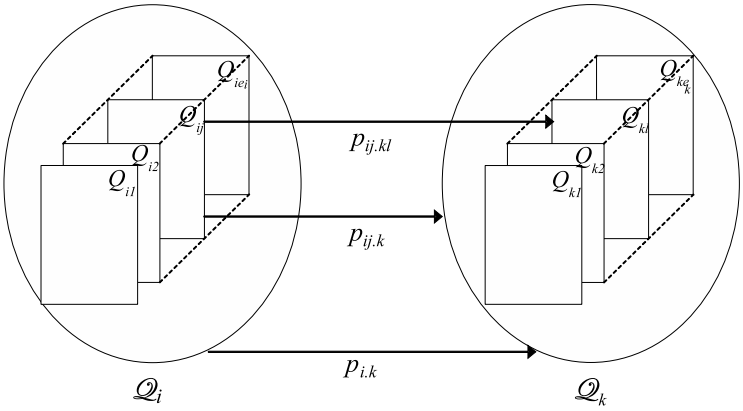
avec $g_i \in \mathcal{G}$ et $t \in \mathcal{T}^2$.

1.3.2 Définition de la chaîne de Markov dans le cas de l'AG

Dans notre contexte, on étudie l'évolution de la population au fur et à mesure des générations. On va donc étudier le processus $\{G_t\}_{t\in\mathcal{T}}$, où G_t est la population obtenue après t application(s) des opérateurs (sélection, croisement, mutation), soit t générations de l'AG. L'espace d'états est donc $\mathcal{G} = \mathcal{Q} = \{Q_{ij}\}$ pour $i = \{1, 2, \ldots, s\}$ et $j = \{1, 2, \ldots, e_i\}$, l'ensemble des populations possibles, de dimension $\mathcal{N} = \sum_{i=1}^{s} e_i$. G_0 est la population initiale. $\mathcal{T}$ représente les temps d'apparition des générations successives, on étudie donc le processus sur $\mathcal{T} = \mathbb{N}$.
De plus, le résultat de l'application des opérateurs ne dépend que de la population courante, on a donc une chaîne de Markov d'ordre 1.
Enfin, puisque la probabilité de mutation évolue au cours du temps, la chaîne de Markov est non homogène.

Probabilités initiales
Nous avons vu (paragraphe 1.5) que l'AG est initialisé avec une population ne contenant que les seuils optimaux pour chaque pic. On a donc $\pi_0(Q_{ij}) > 0$ si Q_{ij} ne contient que des chromosomes n'ayant que des seuils optimaux et $\pi_0(Q_{ij}) = 0$ sinon.

Fig. 1.1 Représentation des probabilités de transitions (sans tenir compte du nombre de générations nécessaires) entre populations et groupes de populations homogènes (au sens de la fitness).



Probabilités de transition

Pour toute génération, on passe d'une population Q_{ij} au temps t à une population Q_{kl} au temps $(t+1)$ avec $(i,k) \in \{1,2,\ldots,s\}^2$, $j \in \{1,2,\ldots,e_i\}$, $l \in \{1,2,\ldots,e_k\}$ et $t \in \mathbb{N}$ avec une probabilité $p_{ij.kl}(t,t+1)$. On note également $p_{ij.k}(t,t+1)$, la probabilité de transition de Q_{ij} au temps t vers toute population de $\mathcal{Q}_k$ au temps $(t+1)$. Ces probabilités sont illustrées dans la Fig. 1.1. On a alors :

$$p_{ij.k}(t,t+1) = \sum_{l=1}^{e_k} p_{ij.kl}(t,t+1), \tag{1.2}$$

et

$$\sum_{k=1}^{s} p_{ij.k}(t,t+1) = 1. \tag{1.3}$$

De plus, $p_{ij.kl}(t,t+n)$ désigne la probabilité d'aboutir à la population Q_{kl} au temps $(t+n)$ à partir de Q_{ij} au temps t, c'est-à-dire en n générations à partir du temps t. On a de même :

$$p_{ij.k}(t,t+n) = \sum_{l=1}^{e_k} p_{ij.kl}(t,t+n).$$

Conséquences de l'étape d'élitisme

Etudions maintenant les conséquences de l'étape d'élitisme sur les probabilités de transition. Grâce à l'application de l'élitisme, le meilleur chromosome (au sens de la fitness) de la population G_t $(t \in \mathbb{N})$ est systématiquement introduit dans la population suivante G_{t+1}. D'où

$$\max_{m=\{1,2,\ldots,M\}} fit(S_m^t) \leq \max_{m=\{1,2,\ldots,M\}} fit(S_m^{t+1}) \tag{1.4}$$

où S_m^t est le $m^{\text{ème}}$ chromosome de la population G_t. Or, d'après la définition 1.2, $fit(G_t) = \max_{m=\{1,2,\ldots,M\}} fit(S_m^t)$, donc l'équation 1.4 implique que $fit(G_t) \leq fit(G_{t+1})$. La conséquence immédiate est la propriété suivante :

Propriété 1.9 *Conséquence de l'élitisme sur les probabilités de transition :*

$$p_{ij.k}(t,t+1) \geq 0 \text{ si } k \leq i, \forall\, t \in \mathbb{N}$$
$$p_{ij.k}(t,t+1) = 0 \text{ si } k > i, \forall\, t \in \mathbb{N}$$

Définition 1.5 *Etat absorbant :*
Un état Q_{ij} est dit absorbant ssi $p_{ij.ij}(t,t+1) = 1$, $\forall\, t \in \mathbb{N}^2$, c'est-à-dire (d'après l'équation 1.3) $p_{ij.kl}(t,t+1) = 0$ si $i \neq k$ et $j \neq l$.

Or d'après la propriété 1.9, $p_{1j.kl}(t,t+1) = 0$, $\forall\, k > 1$, donc $p_{1j.1}(t,t+1) = 1$. L'ensemble $\mathcal{Q}_1$ est absorbant : si on trouve une population $G_t \in \mathcal{Q}_1$ alors $G_{t+k} \in \mathcal{Q}_1$, $\forall\, k \in \mathbb{N}$. On peut évidemment aboutir à plusieurs populations Q_{1j} où $j \in \{1, 2, \ldots, e_1\}$ mais elles seront toutes optimales au sens de la fitness et l'AG aura donc convergé.

1.3.3 Convergence de l'AG

Définition 1.6 *: Convergence de l'AG*
L'AG a convergé en G générations si et seulement si :

$$\exists m \in \{1, 2, \ldots, M\} \text{ et } S_m \in Q^{(G)} \text{ tel que } fit(S_m) = F_1.$$

où $Q^{(G)}$ est la $G^{ème}$ génération obtenue et S_m est le $m^{ème}$ chromosome de la population $Q^{(G)}$.

Pour commencer, nous allons démontrer la convergence de l'AG dans le cas homogène (les probabilités de transition ne dépendent pas du temps). Ainsi, pour alléger les écritures, nous écrirons $p_{ij.kl}$ pour $p_{ij.kl}(t,t+1)$ et $p^{(n)}_{ij.kl}$ pour $p_{ij.kl}(t,t+n)$.
Nous montrerons dans un premier temps que notre AG permet de passer de toute population à toute autre population de fitness supérieure ou égale en un nombre fini de générations avec une probabilité non nulle. Nous utiliserons ensuite cette propriété pour démontrer que :

$$\lim_{n\to\infty} p^{(n)}_{ij.1} = 1,\ \forall i \in \{1, 2, \ldots, s\},\ \forall j \in \{1, 2, \ldots, e_i\}. \tag{1.5}$$

Montrons donc tout d'abord la propriété suivante :

Propriété 1.10

$$\exists N \in \mathbb{N} \text{ tel que } p^{(n)}_{ij.kl} > 0, \text{ pour } n \geq N \text{ et } i \geq k.$$

En effet, si tel n'est pas le cas, il sera impossible de garantir que l'AG *tombe* au bout d'un certain temps dans $\mathcal{Q}_1$ et donc converge. Or, on sait que $G_t \in \mathcal{Q}_i$ si et seulement si le (ou les) meilleur(s) chromosome(s) de G_t au sens de la fitness ont pour valeur de fitness F_i. Donc démontrer la propriété 1.10 est équivalent à montrer qu'il est possible de passer de n'importe quel chromosome à n'importe quel autre en un nombre fini de générations, en particulier de $S_1 \in \mathcal{S}_i$ à $S_2 \in \mathcal{S}_j$ quand $i > j$.

Démonstration :
On peut commencer par envisager le pire cas, c'est-à-dire qu'on veut passer du chromosome S_1 à S_2 sachant qu'il n'y a aucun élément commun entre les deux. Supposons également,

pour simplifier, que l'on ne dispose que de la mutation. Alors, d'après la définition de la mutation, nous avons trois possibilités : déplacer un seuil, supprimer un pic et ajouter un pic. Ainsi, si S_1 comporte K_1 pics et S_2, K_2 pics, il faut, au pire, supprimer K_1 pics, ajouter K_2 pics et déplacer K_2 seuils. Il faut donc au moins K_1+2K_2 générations. A chaque étape, la probabilité de mutation est non nulle donc, en K_1+2K_2 générations, la probabilité d'obtenir S_2 à partir de S_1 est non nulle. Evidemment, cette probabilité peut être très faible (il faut choisir de faire une mutation et appliquer la bonne mutation) mais elle est non nulle. On peut également trouver un majorant de $K_1 + 2K_2$, en effet, $K_1 \leq N_m$ et $K_2 \leq N_m$ donc $K_1 + 2K_2 \leq 3N_m$. Posons $N = 3N_m$.
Par conséquent, on a bien :

$$p_{ij.k}^{(n)} > 0 \text{ si } k \leq i \text{ et } n \geq N. \square$$

Démontrons maintenant par récurrence la propriété suivante :

Propriété 1.11

$$\sum_{k=2}^{s} p_{ij.k}^{(cN+1)} \leq \delta^c(1-p_{ij.1}), \forall i \in \{1,2,\ldots,s\}, \forall j \in \{1,2,\ldots,e_i\}, \text{ où} \delta = \max_{i,j}(1-p_{ij.1}^{(N)}) \, etc \in \mathbb{N}.$$

Or, nous venons de voir que $p_{ij.1}^{(N)} > 0$ donc $\delta < 1$.

Démonstration :

$$\begin{aligned}
\sum_{k=2}^{s} p_{ij.k}^{(N+1)} &= \sum_{k=2}^{s}\sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1} p_{i_1j_1.k}^{(N)} \\
&= \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1} \sum_{k=2}^{s} p_{i_1j_1.k}^{(N)} \\
&= \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(1 - p_{i_1j_1.1}^{(N)}) \\
&\leq \delta \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1} \\
&= \delta \sum_{i_1=2}^{s} p_{ij.i_1} \\
&= \delta(1 - p_{ij.1})
\end{aligned}$$

La propriété 1.11 est donc vérifiée pour $c = 1$. Supposons maintenant que $\sum_{k=2}^{s} p_{ij.k}^{(cN+1)} \leq$

$\delta^c(1-p_{ij.1})$. Alors,

$$\begin{aligned}
\sum_{k=2}^{s} p_{ij.k}^{((c+1)N+1)} &= \sum_{k=2}^{s}\sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}^{(N)} p_{i_1j_1.k}^{(cN+1)} \\
&= \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}^{(N)} \sum_{k=2}^{s} p_{i_1j_1.k}^{(cN+1)} \\
&\leq \delta^c \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}^{(N)}(1-p_{ij.1}) \text{ d'après l'hypothèse de récurrence} \\
&= \delta^c \sum_{i_1=2}^{s} p_{ij.i_1}^{(N)}(1-p_{ij.1}) \\
&\leq \delta^{c+1}(1-p_{ij.1})
\end{aligned}$$

Ainsi, si la propriété 1.11 est vérifiée pour c alors elle l'est pour $(c+1)$. On a donc bien montré que $\sum_{k=2}^{s} p_{ij.k}^{(cN+1)} \leq \delta^c(1-p_{ij.1})$, $\forall c \in \mathbb{N}$ □.

Il en découle que :

$$\lim_{c\to\infty}\{\sum_{k=2}^{s} p_{ij.k}^{(cN+1)}\} = 0,$$

donc

$$\lim_{c\to\infty} p_{ij.1}^{(cN+1)} = 1.$$

On a bien convergence de l'AG dans le cas homogène.

1.3.4 Généralisation au cas non homogène

Nous allons maintenant nous intéresser au cas non homogène, c'est-à-dire le cas où les probabilités de transition d'un état à l'autre varient en fonction du temps. On considère donc réellement $p_{ij.kl}(t,t+n)$ comme une fonction du temps. Cette évolution au cours du temps correspond au cas que nous avons évoqué à plusieurs reprises au cours de ce travail, celui où le taux de mutation, $p_m(t)$ évolue au cours des générations. Le plus souvent cela consistera à avoir un taux de mutation élevé en début d'algorithme (pour permettre une meilleure exploration de l'espace des solutions) puis une diminution de ce taux (pour permettre à l'algorithme de converger). Pour cela, nous allons adopter un raisonnement analogue à celui présenté pour le cas homogène (taux de mutation constant). Nous allons tout d'abord montrer par récurrence une relation analogue à la propriété (1.11) :

Propriété 1.12

$$\sum_{k=2}^{s} p_{ij.k}(t,t+cN+1) \leq \delta^c(1-p_{ij.1}(t,t+1)),\ \forall i \in \{1,2,\ldots,s\},\ \forall j \in \{1,2,\ldots,e_i\},$$

où $\delta = \max_{i,j,t}(1-p_{ij.1}(t,t+N)$ *et* $c \in \mathbb{N}$.

Par rapport au cas homogène, nous avons une difficulté supplémentaire concernant le signe de δ. En effet, nous avons montré précédemment que $p_{ij.1}(t, t+N) > 0$ mais comme nous considérons une série ($t \to \infty$), on ne peut conclure immédiatement que $\min_{i,j,t} p_{ij.1}(t, t+N) > 0$.
Dans le cas homogène, la série des probabilités de transition est une constante, elle est donc divergente. Alors, d'après le lemme de Borel-Cantelli, quand la somme des probabilités d'apparition des termes d'une suite est infinie, tous les états sont visités avec une probabilité de 1. En particulier, on passera dans le sous-ensemble des états absorbants. On peut donc, dans le cas homogène, être assuré de la convergence de l'algorithme.
Dans le cas non homogène, on ne peut pas appliquer directement le lemme de Borel-Cantelli. Cependant, en ajoutant une hypothèse peu contraignante sur la fonction $p_m(t)$, nous allons pouvoir appliquer le même raisonnement. Cette hypothèse est la suivante :

Hypotèse 1.1 ***Contrainte sur la fonction*** $p_m(t)$ *:*
La fonction $p_m(t)$ *admet un minimum* $0 < \mu < 1$*.*

Cette hypothèse n'est pas très forte en pratique. En effet, si l'opérateur de mutation disparaît, on n'est plus vraiment dans le cas d'un algorithme génétique. De plus, nous avons vu que c'est le seul opérateur qui permet une vraie exploration de l'espace des solutions. Si on n'a plus de mutations, alors, seuls les allèles présents dans la population au moment où le taux de mutation devient nul pourront intervenir dans les solutions des populations à venir. Ce n'est donc pas un cas souhaitable. Voyons maintenant les implications de cette hypothèse pour notre problème.
Les probabilités de transition dépendent essentiellement du taux de mutation et du taux de croisement (du fait de l'élitisme et de la définition de nos états, la pression de sélection n'intervient pas directement). Or, le taux de croisement est constant, donc seul le taux de mutation est variable dans les probabilités de transition. On peut donc écrire $p_{ij.1}(t, t+N)$ comme une fonction $f(p_m(t))$. Or, nous savons que, plus le taux de mutation est élevé, plus on explore l'espace des solutions largement. De plus, l'élitisme empêche toutes les transitions d'un groupe $\mathcal{Q}_k$ à un groupe $\mathcal{Q}_l$ si $l > k$. Donc, en augmentant le taux de mutation, on diminue la probabilité de rester dans le même groupe (sauf pour $\mathcal{Q}_1$) et on augmente la probabilité d'atteindre un groupe de fitness supérieure. On a donc plus de chances d'arriver dans $\mathcal{Q}_1$. Donc, $p_{ij.1}(t, t+N)$ est une fonction croissante de $p_m(t)$. On en déduit que

$$p_m \geq \mu \Rightarrow p_{ij.1} \geq f(\mu).$$

Ensuite, si on construit une série constante : $u(t) = f(\mu)$ pour tout $t \in \mathbb{N}$ alors,

$$\forall t \in \mathbb{N},\ \ p_{ij.1}(t, t+N) \geq u(t).$$

Or, $u(t)$ est une série constante donc divergente, $p_{ij.1}(t, t+N)$ est donc minorée par une série divergente, donc $p_{ij.1}(t, t+N)$ est divergente. On peut donc appliquer au cas non homogène les raisonnements du cas homogène, en particulier, dire que $p_{ij.1}(t, t+N) > 0$ donc $\delta < 1$.
Nous pouvons maintenant passer à la démonstration de la propriété 1.12.

Démonstration

$$\begin{aligned}
\sum_{k=2}^{s} p_{ij.k}(t,t+N+1) &= \sum_{k=2}^{s}\sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(t,t+1)p_{i_1j_1.k}(t+1,t+N+1) \\
&= \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(t,t+1)\sum_{k=2}^{s} p_{i_1j_1.k}(t+1,t+N+1) \\
&= \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(t,t+1)(1-p_{i_1j_1.1}(t+1,t+N+1)) \\
&\leq \delta\sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(t,t+1) \\
&= \delta\sum_{i_1=2}^{s} p_{ij.i_1}(t,t+1) \\
&= \delta(1-p_{ij.1}(t,t+1))
\end{aligned}$$

La propriété 1.12 est donc vérifiée pour $c=1$. Supposons maintenant que $\sum_{k=2}^{s} p_{ij.1}(t,t+cN+1) \leq \delta^c(1-p_{ij.1}(t,t+1))$ (hypothèse de récurrence notée HR). Alors,

$$\sum_{k=2}^{s} p_{ij.k}(t,t+((c+1)N+1)$$

$$= \sum_{k=2}^{s}\sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(t,t+cN+1)p_{i_1j_1.k}(t+cN+1,t+((c+1)N+1)$$

$$= \sum_{i_1=2}^{s}\sum_{j_1=1}^{e_{i_1}} p_{ij.i_1j_1}(t,t+cN+1)\sum_{k=2}^{s} p_{i_1j_1.k}(t+cN+1,t+((c+1)N+1)$$

$$\leq \delta^c\sum_{k=2}^{s} p_{i_1j_1.k}(t+cN+1,t+((c+1)N+1) \text{ d'après l'HR}$$

$$\leq \delta^{c+1}(1-p_{ij.1}(t,t+1))$$

Ainsi, si la propriété 1.12 est vraie pour c alors elle l'est pour $(c+1)$. On a donc bien montré que $\sum_{k=2}^{s} p_{ij.k}(t,t+cN+1) \leq \delta^c(1-p_{ij.1}(t,t+1))$, $\forall(c,t)\in\mathbb{N}^2$ $\square$.
Il vient donc que

$$\lim_{c\to\infty}\{\sum_{k=2}^{s} p_{ij.k}(t,t+cN+1)\} = 0$$

donc

$$\lim_{c\to\infty} p_{ij.1}(t,t+cN+1) = 1.$$

On a également convergence de l'AG dans le cas non homogène.

1.3.5 Influence de l'opérateur de croisement sur la vitesse de convergence

Nous avons considéré dans un premier temps, pour simplifier, qu'on ne faisait intervenir que l'opérateur de mutation qui ne permet qu'un changement à la fois. A partir de cette simplification, nous avons pu donner un nombre minimum de générations nécessaires à la convergence de l'algorithme qui était de $N = 3N_m$. Cependant, l'introduction de l'opérateur de croisement permet de nettement réduire l'ordre de grandeur de N. En effet, il peut permettre de combiner des caractéristiques intéressantes apparues simultanément dans différents chromosomes de la population par mutation.
Envisageons à nouveau le pire des cas. Notre objectif est d'obtenir un chromosome de $\mathcal{S}_1$. Choisissons donc un chromosome *objectif* qui soit un chromosome de $\mathcal{S}_1$. Supposons que dans la population initiale, il n'y ait aucun des pics constituant les chromosomes de $\mathcal{S}_1$ mais que tous les chromosomes aient N_m pics. Choisissons un chromosome *objectif* qui soit un chromosome de $\mathcal{S}_1$. Il est alors nécessaire dans une première étape de supprimer N_m pics dans la population, ceci peut être effectué en une seule génération en appliquant la mutation *enlever un pic* à au moins N_m chromosomes. Ensuite, dans la génération suivante, on peut, toujours par l'opérateur de mutation, faire apparaître les N_m pics nécessaires (toujours dans des chromosomes différents). Une autre génération est nécessaire pour obtenir les seuils *objectifs*.
On dispose, à ce stade, de tous les éléments du chromosome *objectif*, il ne reste plus qu'à effectuer autant de croisements que nécessaire, sachant que l'opérateur de croisement employé permet de croiser entre 1 et N_m pics. Cependant, un croisement ne concerne que deux chromosomes donc, à chaque génération, on divise au maximum par 2 le nombre de chromosomes contenant les bons caractères (pics et seuils). En fait, si à une étape, les bons caractères sont répartis dans $2n$ chromosomes, alors ils seront répartis dans n chromosomes à l'étape suivante. Par contre, s'ils sont répartis dans $2n+1$ chromosomes à une étape, l'un des chromosomes ne pourra pas être croisé et les bons caractères seront répartis dans $(n+1)$ chromosomes.
Prenons un exemple pour lequel $N_m = 10$. A l'étape 1, les caractères *objectifs* sont répartis dans 10 chromosomes. A l'étape 2, ils sont croisés deux à deux et on retrouve donc les bons caractères dans 5 chromosomes. A l'étape 3, 4 des chromosomes peuvent être croisés deux par deux pour constituer deux nouveaux chromosomes et il en reste un de côté, on a donc une répartition dans 3 chromosomes. De la même façon, à l'étape 4, un seul croisement est possible ce qui aboutit à une répartition en 2 chromosomes qui sont croisés dans l'étape 5 pour donner le chromosome *objectif*. Pour arriver à rassembler 10 caractères, il faut donc 4 étapes.
Ainsi, si le nombre de caractères à combiner, a, peut s'exprimer sous la forme $a = 2^n$, il faudra $n = log_2(a)$ étapes de croisements. Si $2^n < a \leq 2^{n+1}$ alors il faudra $n+1$ étapes de croisement. Soit $C(N_m)$ le nombre d'étapes de croisements nécessaires pour combiner N_m caractères, alors :

$$\begin{aligned} C(N_m) &= [log_2(N_m)] \qquad \text{si } log_2(N_m) = [log_2(N_m)] \\ &= [log_2(N_m)] + 1 \ \text{ si } log_2(N_m) > [log_2(N_m)] \end{aligned}$$

Donc, au total, en tenant compte de l'étape de croisement, il faut $N = 3 + C(N_m)$ générations pour avoir

$$p_{ij.k}^{(n)} > 0 \text{ si } k \leq i \text{ et } n \geq N.$$

L'ajout du croisement a donc de fortes conséquences, si $N_m = 10$ alors $N = 30$ si on ne tient pas compte du croisement alors que $N = 7$ en en tenant compte. De façon plus générale, si on double le nombre maximum de pics, alors on double également N dans le cas sans croisement alors que N n'augmente que de 1 en tenant compte des croisements.

Chapitre 2

Construction d'un critère de pseudo-convergence

2.1 Introduction

Les résultats de convergence tels que celui précédemment démontré sont très importants et tout à fait nécessaires pour comprendre le fonctionnement des AG et justifier la démarche proposée. Cependant, d'un point de vue pratique, ces résultats sont rassurants mais très peu utiles. En effet, la convergence de l'AG est assurée mais la vitesse de convergence est inconnue. En pratique, ce qui est nécessaire, c'est un critère d'arrêt pour l'AG. Nous avons déjà vu qu'il était possible d'arrêter l'algorithme au bout d'un certain temps de calcul ou d'un certain nombre de générations tous deux prédéfinis, on peut aussi choisir un temps pendant lequel la fitness moyenne dans la population n'évolue pas beaucoup. Ce dernier cas semble plus intéressant car il prend réellement en compte la convergence observée de la population. Toutefois, en général, le taux de mutation n'est jamais nul et il y a donc toujours des individus dans la population qui mutent et sont ainsi susceptibles de dévier de l'optimum. La moyenne de la fitness dans la population s'en trouvera nécessairement modifiée. Il est donc indispensable de définir une marge de tolérance à l'intérieur de laquelle on considère que la moyenne de la population n'a pas évolué. Cette marge doit évidemment évoluer en fonction de l'amplitude de la fitness et donc du problème. Ce qui implique de nouveaux réglages dès que l'on change de problématique.

Afin de remédier à ces inconvénients, nous avons essayé de mettre en place un nouveau critère d'arrêt. Il est important de noter que ce critère n'a pas la prétention de repérer la présence de l'optimum global. On pourrait dire qu'il s'agit d'un critère de pseudo-convergence. L'objectif d'un tel critère est de réaliser un compromis entre les résultats purement théoriques inutilisables en pratique et les critères d'arrêt usuels qui n'ont aucun fondement théorique. Il est bien évident que, dans le cas d'une heuristique comme les AG, il est impossible, en pratique, de savoir si l'on a obtenu l'optimum global. Mais, on peut essayer de s'en rapprocher au maximum. Pour construire ce critère, nous sommes partis du fait qu'une solution de bonne qualité (localement ou globalement optimale) *colonise* petit à petit la population. Ce phénomène est assez intuitif étant donné que l'élitisme et l'opérateur de sélection tendent à maintenir la meilleure solution et à favoriser son apparition dans la génération suivante. Une étude théorique sur ce sujet a été menée par Cerf (1996a). L'introduction de l'élitisme nous place dans un contexte différent pour lequel nous allons modéliser la *colonisation* de la

population par les optima locaux successifs. Pour pouvoir réaliser cette modélisation, nous allons devoir procéder à quelques simplifications. On disposera donc d'un modèle qui ne reflètera pas totalement la réalité mais qui nous permettra de donner une idée de l'assise théorique du critère de convergence que nous allons construire.

Avant de rentrer dans les détails du critère et de sa modélisation, nous allons présenter la petite simulation qui servira d'illustration, elle permettra de vérifier la pertinence de notre critère. Pour cela, nous allons construire deux AG qui permettent de réaliser une classification non supervisée. Bien que les deux algorithmes auront le même objectif, leurs méthodes d'optimisation seront très différentes et nous permettront de tester l'adaptabilité du critère.

2.2 Simulation

2.2.1 Construction des AG

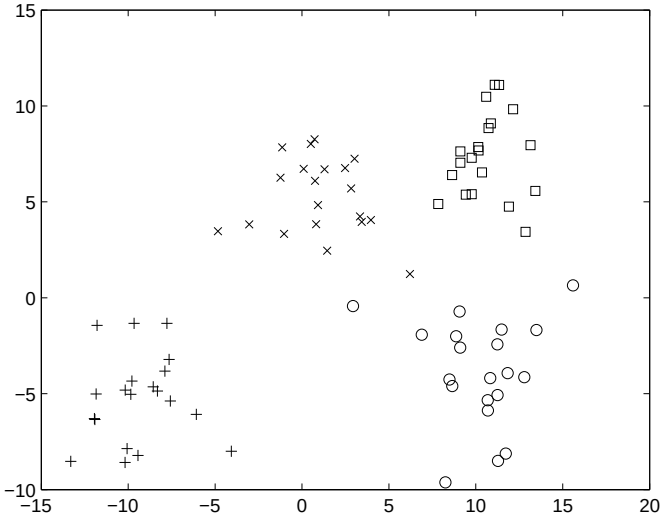
2.2.1.1 Première approche

On suppose que l'on dispose d'un tableau de données comportant n individus et p variables. La première méthode pour réaliser une classification non supervisée est la plus directe car elle portera sur les affectations des n individus. Une solution comptera autant de loci que d'individus à classer et chacun des individus se verra affecter un groupe d'appartenance. Le nombre de groupes, k, pour chaque solution est défini aléatoirement lors de la génération de la première population, il est généré entre 2 et N_{max} qui est fixé préalablement. Ensuite, on affecte à chacun des loci de la solution un nombre entre 1 et k. Pour la mutation, on définit un taux de mutation, p_m qui est appliqué à chaque solution. Une mutation peut consister à ajouter, supprimer un groupe ou à modifier une ou plusieurs affectations. Quant au croisement, nous avons appliqué le *croisement uniforme* comme décrit au paragraphe 1.4. Enfin, la sélection utilise les rangs, comme dans toutes nos applications, elle porte sur la fitness qui tient compte de la somme des distances intra-groupes moyennes et du nombre de groupes.

2.2.1.2 Seconde approche

La deuxième méthode que nous proposons est plus proche de l'algorithme k-means (Hastie *et al.*, 2001). En effet, au lieu des affectations, nous allons optimiser les positions des centres de gravité des groupes. Nous fixerons, comme précédemment, un nombre maximum de groupes, N_{max}. Cela nous permettra d'avoir une longueur de codage constante. En effet, une solution contiendra les positions des centres des groupes soit une longueur de $N_{max} \times p$ avec uniquement $k \times p$ loci contenant des positions si la solution courante contient k groupes. Pour l'opérateur de mutation, on appliquera à nouveau un taux de p_m à chaque solution en autorisant suppression, addition et déplacement des centres. Le croisement est à nouveau uniforme mais on choisit de croiser un centre complet (soit p coordonnées). Enfin, pour la fitness, on calcule les affectations (au centre le plus proche) et la fitness est définie comme pour la première méthode. Notons qu'une fois les affectations réalisées, les centres sont recalculés comme étant les centres de gravité des nouveaux groupes.

Fig. 2.1 Représentation des deux premières variables de la simulation retenue pour les AG permettant la classification non supervisée d'un jeu de données. Chaque symbole représente un groupe.



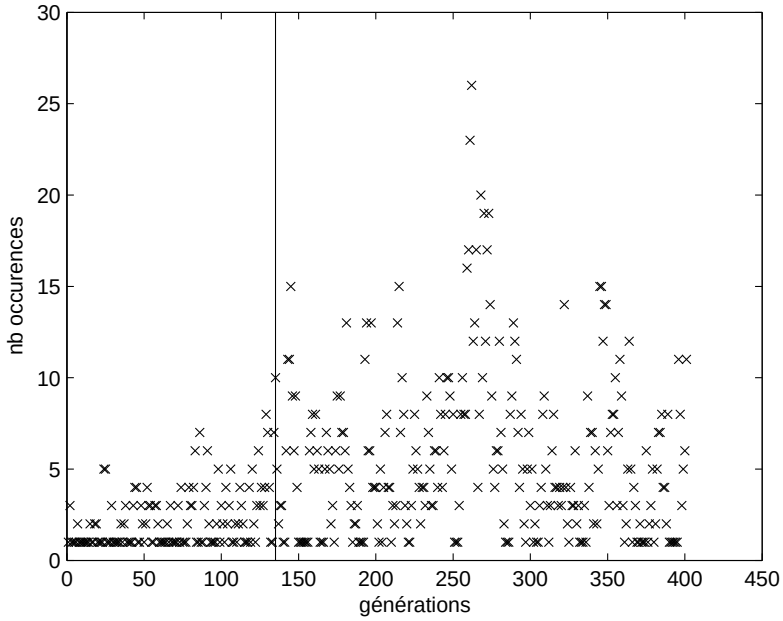
2.2.2 Construction d'un jeu de données simulé

Pour tester les deux algorithmes précédemment décrits, nous avons simulé cinq variables, dont deux qui permettent de positionner les groupes et les trois autres qui sont des variables de bruit. Ces dernières variables permettent d'introduire une difficulté de plus. Pour les variables de position des groupes, nous avons simulé quatre groupes de vingt observations chacun, par des lois normales univariées dont la moyenne reflète les positions des centres des groupes et la variance leur étendue. Finalement, pour la simulation retenue, on obtient pour ces deux variables de position la répartition de la Fig. 2.1. On remarque qu'il existe un groupe bien isolé alors que les trois autres sont plus proches. Pour les variables de bruit, nous avons simulé des lois uniformes univariées sur l'intervalle $[-5, 5]$.

2.3 Construction du critère

Nous avons déjà évoqué que le critère d'arrêt allait se baser sur le nombre d'occurrences de la meilleure solution courante au fur et à mesure des générations. Notons bien ici, qu'il s'agit de la meilleure solution courante, c'est-à-dire la meilleure dans la génération courante et non de la meilleure solution globale. Pour nous convaincre de l'intérêt de ce paramètre, observons son évolution au cours des générations dans la Fig. 2.2. On voit immédiatement que le nombre d'occurrences de la meilleure solution courante augmente très rapidement une fois que la meilleure solution globale est apparue. Il semble donc tout à fait approprié de baser un critère d'arrêt sur ce paramètre. D'après la Fig. 2.2, on pourrait penser à utiliser une méthode de détection de rupture mais nous allons construire une méthode qui est plus adaptée car elle va réellement prendre en compte l'ensemble des mécanismes inhérents à l'AG.

Fig. 2.2 Evolution du nombre d'occurrences de la meilleure solution courante au cours des 400 premières générations pour les données simulées. Le trait vertical indique la première apparition dans la population courante de la solution **globalement** optimale.



2.3.1 Construction de $\{Z_n\}$

Intéressons-nous dans un premier temps au processus $\{Z_n\}$:

$$Z_n = \{\textit{Nombre d'occurrences de la meilleure solution courante dans la génération } n\}.$$

Il serait très intéressant de pouvoir considérer ce processus comme une chaîne de Markov et nous avons vu dans la section précédente que la population peut être modélisée par une chaîne de Markov d'ordre 1. Malheureusement, d'une manière générale, le nombre d'occurrences de la meilleure solution courante ne dépend pas que du nombre d'occurrences de la meilleure solution dans la génération précédente, il dépend également de la nature des autres solutions présentes dans la population précédente. En effet, par le biais des mutations et des croisements, ces solutions peuvent évoluer pour atteindre la même fitness ou même une fitness supérieure. Profitons-en pour noter que l'étude de ce processus nécessitera la considération de deux cas différents :

- la meilleure solution courante est la même que dans la génération précédente,
- une nouvelle meilleure solution courante vient d'apparaître.

Mais dans les deux cas, la structure de la population dans son ensemble (et pas seulement le nombre d'occurrences de la meilleure solution courante) aura une influence sur le nombre d'occurrences de la meilleure solution locale dans la génération suivante.
Cependant, afin de pouvoir approcher d'une manière théorique le contexte dans lequel se situe notre critère, nous allons faire deux hypothèses qui nous permettront de considérer le processus $\{Z_n\}$ comme une chaîne de Markov. Pour cela, nous allons noter T_n la variable aléatoire qui vaut 0 si on a changé de meilleure solution courante entre la génération $n-1$ et la génération n et 1 sinon.

Hypotèse 2.1 ***Hypothèse 1 : loi de*** T_n
$\mathbf{Pr}[T_n = 1] = \varphi$ *et* $\mathbf{Pr}[T_n = 0] = 1 - \varphi$*, avec* $\varphi \in [0, 1[$*,*

Examinons le sens de cette première hypothèse. En fixant cette loi pour T_n, on indique que la probabilité de changer d'optimum n'évolue pas au cours des générations. En fait, il est évident que, en réalité, cette probabilité change. Quand la population a évolué depuis un certain nombre de générations et que des caractéristiques intéressantes sont apparues dans la population, il est clair que la probabilité d'obtenir un meilleur optimum augmente. A l'opposé, lorsque l'on est à l'optimum global, on n'a plus aucune chance d'améliorer l'optimum, $\mathbf{Pr}[T_n = 0] = 0$. Par contre, au début de l'algorithme, si on a réalisé une initialisation aléatoire, on ne sait pas *a priori* si elle va donner lieu à une meilleure solution ou non dans la génération suivante. Dans un tel cas, on pourrait poser $\mathbf{Pr}[T_n = 1] = \mathbf{Pr}[T_n = 0] = 0.5$. Cependant, notre critère va s'intéresser aux transitions de Z_n dans le cas où la chaîne n'a pas convergé et quand le comportement de la chaîne s'éloignera trop du comportement modèle, alors on considérera que l'on a convergé. On veut donc modéliser les transitions de Z_n sous l'hypothèse de non convergence de l'algorithme et on ne considérera donc que des valeurs $\mathbf{Pr}[T_n = 0] > 0$.

Hypotèse 2.2 ***Hypothèse 2 :***
On néglige la probabilité d'apparition de nouvelles occurrences de la meilleure solution courante parmi les solutions autres que celles qui sont déjà localement optimales.

Par cette hypothèse, on estime que c'est uniquement l'opérateur de sélection qui permet la *colonisation* de la population par les optima successifs. On néglige le fait que les opérateurs de croisement et de mutation peuvent faire apparaître la même solution ou une solution différente mais de même fitness. Comme nous le verrons par la suite, cette hypothèse est tout à fait acceptable si on se trouve dans un problème où la fonction de fitness peut prendre de très nombreuses valeurs et si l'espace des solutions est très grand. Il est évident que si le fitness peut prendre dix valeurs ou si l'on n'optimise que deux paramètres qui peuvent prendre vingt valeurs chacun, cette hypothèse ne pourra pas être vérifiée. Mais, dans une grande partie des domaines d'application des AG, ce dernier cas n'apparaît pas. En effet, on a plus tendance à utiliser les AG pour des problèmes ayant un très grand nombre de solutions possibles. Nous verrons par la suite quelques exemples de restriction.

Si l'on pose les deux hypothèses précédentes, alors on n'a plus besoin de connaître la structure du reste de la population (hors solutions localement optimales), on a simplement besoin du Z_{n-1} pour prédire le nombre d'occurences de la meilleure solution courante pour la génération suivante. Nous pouvons alors considérer Z_n comme une chaîne de Markov d'ordre 1 dont nous allons modéliser les transitions. Les conséquences et les restrictions dues aux hypothèses présentées seront abordées dans la suite.

Dans l'éventualité où la meilleure solution ne change pas, le nombre d'occurrences dans la génération $n + 1$ ne dépend que du nombre d'occurrences dans la génération n. En effet, d'après l'hypothèse 2.2, les meilleures solutions de la génération $n + 1$ proviennent uniquement de la sélection des meilleures solutions qui étaient présentes dans la génération n et qui n'ont pas été détruites par l'application des opérateurs de mutation et de croisement.
Pour le cas où on change de meilleure solution, on posera $Z_n = 1$ et ce, pour deux raisons.

A terme, nous allons observer la somme d'une suite de réalisations de Z_n et nous voulons avoir des sommes importantes quand on n'a pas changé de solution optimale depuis de nombreuses générations. Quand on vient de changer de solution optimale, on veut donc donner une valeur faible à Z_n. Cette contrainte n'est, en outre, pas aberrante puisque la probabilité pour qu'une meilleure solution apparaisse plus d'une fois dans la même population (lors des étapes de mutation et croisement) est très faible dans la majorité des problèmes.

L'espace d'état de Z_n est naturellement $\{1, 2, \ldots, T_{pop}\}$ où T_{pop} est toujours la taille de la population. Pour ce qui est de l'état initial, on pose naturellement $Z_0 = 1$. Etudions maintenant l'ensemble des probabilités $\pi_n(k, l)$ où

$$\pi_n(k, l) = \mathbf{Pr}[Z_n = k | Z_{n-1} = l].$$

Pour calculer ces probabilités, nous allons devoir considérer successivement plusieurs conditionnements.

2.3.1.1 Conditionnement 1

Le premier conditionnement porte sur ce qui vient d'être évoqué, à savoir le changement ou non de meilleure solution courante. On peut noter ici que, grâce à l'élitisme (dont nous avons vu l'importance pour la convergence), un changement de meilleure solution ne peut se faire que dans le sens d'une amélioration. Il ne peut y avoir de détérioration de la meilleure solution. On peut alors décomposer $\pi_n(k, l)$ comme suit :

$$\pi_n(k, l) = \mathbf{Pr}[Z_n = k | Z_{n-1} = l, T_n = 0]\mathbf{Pr}[T_n = 0) + \mathbf{Pr}[Z_n = k | Z_{n-1} = l, T_n = 1]\mathbf{Pr}[T_n = 1]$$

Or, nous avons posé $Z_n = 1$ si on change de meilleure solution. Considérons donc, dans un premier temps, $\pi_n(1, l)$:

$$\begin{aligned} \pi_n(1, l) &= \mathbf{Pr}[Z_n = 1 | Z_{n-1} = l, T_n = 0] \times \mathbf{Pr}[T_n = 0] + \mathbf{Pr}[Z_n = 1 | Z_{n-1} = l, T_n = 1] \\ &\quad \times \mathbf{Pr}[T_n = 1] \\ &= 1 \times \mathbf{Pr}[T_n = 0] + \mathbf{Pr}[Z_n = 1 | Z_{n-1} = l, T_n = 1] \times \mathbf{Pr}[T_n = 1]. \end{aligned}$$

Intéressons-nous maintenant à $\pi_n(k, l)$ pour $k \neq 1$. On peut l'écrire de la même façon que $\pi_n(1, l)$:

$$\begin{aligned} \pi_n(k, l) &= \mathbf{Pr}[Z_n = k | Z_{n-1} = l, T_n = 0] \times \mathbf{Pr}[T_n = 0] + \mathbf{Pr}[Z_n = k | Z_{n-1} = l, T_n = 1] \\ &\quad \times \mathbf{Pr}[T_n = 1] \\ &= 0 \times \mathbf{Pr}[T_n = 0] + \mathbf{Pr}[Z_n = 1 | Z_{n-1} = l, T_n = 1] \times \mathbf{Pr}[T_n = 1]. \end{aligned}$$

Ici, le terme en $T_n = 0$ disparaît puisque quand $T_n = 0$, Z_n prend nécessairement la valeur 1 quand on change de meilleure solution.
Concernant le terme où $T_n = 1$, on peut raisonner de la même façon $\forall k \in \{1, \ldots, T_{pop}\}$.

2.3.1.2 Conditionnement 2

Le nombre d'occurrences de la meilleure solution dans la génération n dépend, dans un premier temps, du nombre de meilleures solutions qui restaient après application des opérateurs de mutation et de croisement à la génération $n-1$. Pour en tenir compte, nous allons introduire une nouvelle variable aléatoire, Z_n^{mc} qui compte le nombre de meilleures solutions restant après application de la mutation et du croisement à la génération n. Si $Z_n = l$ alors on considère que $Z_n^{mc} \in \{0, 1, \ldots, l\}$. On oscille entre deux cas extrêmes, celui où aucune solution localement optimale n'a été détruite, et celui où toutes ces solutions ont été détruites. On décompose alors :

$$\mathbf{Pr}[Z_n = k | Z_{n-1} = l, T = 1] = \sum_{j=1}^{l} \mathbf{Pr}[Z_n = k | Z_{n-1} = l, T = 1, Z_{n-1}^{mc} = j] \mathbf{Pr}[Z_{n-1}^{mc} = j].$$

Calculons tout d'abord $\mathbf{Pr}[Z_{n-1}^{mc} = j]$. On considère p_m et p_c les probabilités respectives de mutation et de croisement pour une solution. On considère que ces probabilités sont constantes au cours du temps. Rappelons que, contrairement à la définition classique, notre probabilité de mutation s'applique à la solution et non à chacun de ses loci. Alors, on définit la probabilité p pour une solution de subir au moins un changement :

$$\begin{aligned} p &= \mathbf{Pr}[\textit{mutation} \cup \textit{croisement}] \\ &= \mathbf{Pr}[\textit{mutation}] + \mathbf{Pr}[\textit{croisement}] - \mathbf{Pr}[\textit{mutation} \cap \textit{croisement}] \\ &= p_m + p_c - p_m p_c \end{aligned}$$

Or, nous nous intéressons au nombre de solutions localement optimales qui n'ont pas été modifiées par la mutation et/ou le croisement. Alors, la probabilité, pour une solution, de ne pas subir de modification est $q = 1 - p = 1 - p_m - p_c + p_m p_c$. Finalement, puisqu'on avait l solutions localement optimales dans la génération $n-1$ alors $Z_{n-1}^{mc} = j$ suit une loi binomiale :

$$\mathbf{Pr}[Z_{n-1}^{mc} = j] = \binom{l}{j} q^j (1-q)^{l-j}.$$

Enfin, il nous reste à calculer $\mathbf{Pr}[Z_n = k | Z_{n-1} = l, T = 1, Z_{n-1}^{mc} = j]$. Pour cela, nous avons besoin d'introduire l'effet de l'élitisme. En effet, nous avons vu qu'au cours de l'étape d'élitisme, la meilleure solution de l'étape $n-1$ vient remplacer la pire solution de l'étape n. L'élitisme n'a donc pas d'influence sur $\{Z_n\}$ dans le cas où on change de meilleure solution. Par contre, si la meilleure solution est conservée, il suffira d'avoir sélectionné $k-1$ occurrences de la solution localement optimale pour avoir k occurrences dans la génération n puisque l'élitisme en ajoute une. Il existe cependant un cas particulier, celui où $k = T_{pop}$. Cela peut se produire dans deux cas distincts, soit on avait sélectionné $T_{pop} - 1$ occurrences et l'élitisme en a ajouté une, soit on en avait déjà sélectionné T_{pop} et l'élitisme en a remplacé une.
Intéressons-nous donc à la probabilité de sélectionner la solution localement optimale si elle est présente en j exemplaires. Nous avons vu au paragraphe 1.5 de la partie 1 que la probabilité de sélectionner un individu dépend de son rang, par rapport à la valeur de sa fitness :

$$\mathbf{Pr}[\textit{sélectionner l'individu de rang } r] = \alpha \times r + \beta.$$

Si plusieurs individus ont la même valeur de fitness, le rang que l'on affecte à chacun d'eux est la moyenne des rangs qu'ils auraient s'ils avaient des valeurs légèrement différentes. Dans notre cas, si les j solutions avaient des valeurs légèrement différentes, la meilleure d'entre elles aurait le rang T_{pop}, la solution de valeur immédiatement inférieure aurait le rang $T_{pop}-1$, etc... Le rang affecté à chacune des j occurrences de la solution localement optimale est donc :

$$\begin{aligned}\overline{r} &= \frac{T_{pop}+(T_{pop}-1)+\ldots+(T_{pop}-j+1)}{j}\\ &= \frac{jT_{pop}-(1+2+\ldots+(j-1))}{j}\\ &= T_{pop}-\frac{1}{j}\times\frac{j(j-1)}{2}\\ &= T_{pop}-\frac{j-1}{2}\end{aligned}$$

Donc, la probabilité de sélectionner chacune des occurrences de la meilleure solution courante est :

$$p_j^* = \alpha\left(T_{pop}-\frac{j-1}{2}\right)+\beta,$$

et la probabilité de sélectionner n'importe laquelle des solutions localement optimales est $j\times p_j^*$.
Alors, pour $k<T_{pop}$:

$$\mathbf{Pr}[Z_n=k|Z_{n-1}=l,T=1,Z_{n-1}^{mc}=j]=\binom{T_{pop}}{k-1}(jp_j^*)^{k-1}(1-jp_j^*)^{T_{pop}-k+1},$$

et pour $k=T_{pop}$,

$$\begin{aligned}\mathbf{Pr}[Z_n=T_{pop}|Z_{n-1}=l,T=1,Z_{n-1}^{mc}=j] &= \binom{T_{pop}}{T_{pop}-1}(jp_j^*)^{T_{pop}-1}(1-jp_j^*)+\binom{T_{pop}}{T_{pop}}(jp_j^*)^{T_{pop}},\\ &= T_{pop}(jp_j^*)^{T_{pop}-1}(1-jp_j^*)+(jp_j^*)^{T_{pop}}.\end{aligned}$$

2.3.1.3 Bilan

Finalement, voici le calcul de $\pi_n(k,l)$ pour trois cas :
si $k=1$,

$$\begin{aligned}\pi_n(1,l) &= \mathbf{Pr}[T_n=0]+\left[\sum_{j=0}^{l}\binom{T_{pop}}{0}(jp_j^*)^0(1-jp_j^*)^{T_{pop}}\binom{l}{j}q^j(1-q)^{l-j}\right]\mathbf{Pr}[T_n=1]\\ &= \mathbf{Pr}[T_n=0]+\left[\sum_{j=0}^{l}(1-jp_j^*)^{T_{pop}}\binom{l}{j}q^j(1-q)^{l-j}\right]\mathbf{Pr}[T_n=1]\end{aligned}$$

si $1 < k < T_{pop}$,

$$\pi_n(k,l) = \left[\sum_{j=0}^{l} \binom{T_{pop}}{k-1} (jp_j^*)^{k-1} (1-jp_j^*)^{T_{pop}-k+1} \binom{l}{j} q^j (1-q)^{l-j}\right] \mathbf{Pr}[T_n = 1],$$

et si $k = T_{pop}$,

$$\pi_n(T_{pop},l) = \left[\sum_{j=0}^{l} \left(T_{pop}(jp_j^*)^{T_{pop}-1}(1-jp_j^*) + (jp_j^*)^{T_{pop}}\right) \binom{l}{j} q^j (1-q)^{l-j}\right] \mathbf{Pr}[T_n = 1].$$

A ce stade de l'étude et d'après l'hypothèse 2.2, nous pouvons remplacer $\mathbf{Pr}[T_n = 1]$ et $\mathbf{Pr}[T_n = 0]$ par leur valeur, 0.5.

2.3.2 Construction de $\{S_w\}$

Par observation de la Fig. 2.2, nous pouvons voir que, pour constater la convergence, l'observation d'une seule réalisation de $\{Z_n\}$ n'est pas suffisante. En effet, cette valeur fluctue beaucoup en raison du caractère stochastique de l'algorithme. Pour avoir une mesure plus fiable, nous allons nous baser sur la somme de plusieurs réalisations successives de $\{Z_n\}$ et nous allons former le processus $\{S_w^{(t)}\}$:

$$S_w^{(t)} = \sum_{i=1}^{w} Z_{t+i},$$

c'est-à-dire la somme de w réalisations successives de Z_n. L'espace d'état de $\{S_w^{(t)}\}$ est donc $\{w, w+1, \ldots, wT_{pop}\}$.
Pour nous débarrasser de l'indice t, nous allons faire l'hypothèse suivante :

Hypotèse 2.3 *On suppose que $\{S_w^{(t)}\}$ est stationnaire, c'est-à-dire que ses caractéristiques ne changent pas au cours du temps. Dans ce cas, cela revient à dire que*

$$\mathbf{Pr}[S_w^{(u)} = j] = \mathbf{Pr}[S_w^{(v)} = j],\ \forall (u,v) \in \mathbb{N}^2,$$

On peut donc étudier simplement $S_w = \sum_{i=1}^{w} Z_i$.

Nous allons voir dans la suite, comment nous vérifions la pertinence de cette hypothèse.
Une fois cette hypothèse faite, nous pouvons déterminer la loi de S_w par récurrence sur la loi conjointe de (S_w, Z_w). Nous allons ainsi montrer la propriété suivante :

Propriété 2.1

$$\forall w,\ \mathbf{Pr}[S_{w+1} = s, Z_{w+1} = k] = \sum_{l=1}^{T_{pop}} \mathbf{Pr}[Z_{w+1} = k | Z_w = l] \mathbf{Pr}[S_w = s-k, Z_w = l].$$

Démonstration
Initialisation : $w = 1$:

Dans ce cas, la fenêtre dans laquelle on fait la somme est seulement de 1, on ne regarde qu'une seule réalisation de Z_n. On en déduit donc que :

$$\mathbf{Pr}[S_1 = s, Z_1 = k] = \begin{cases} \mathbf{Pr}[Z_1 = k] \text{ si } s = k \\ 0 \text{ si } s \neq k \end{cases}$$

La loi stationnaire de Z_n peut être déterminée comme indiqué au paragraphe 2.1 par le premier vecteur propre de la matrice Π dont le calcul a été détaillé au paragraphe 2.3.1.

Récurrence : on suppose la propriété 2.1 vraie pour w, alors :

$$\begin{aligned} \mathbf{Pr}[S_{w+1} = s, Z_{w+1} = k] &= \mathbf{Pr}[S_w = s - k, Z_{w+1} = k] \\ &= \sum_{l=1}^{T_{pop}} \mathbf{Pr}[S_w = s - k, Z_{w+1} = k, Z_w = l] \\ &= \sum_{l=1}^{T_{pop}} \mathbf{Pr}[Z_{w+1} = k | S_w = s - k, Z_w = l] \mathbf{Pr}[S_w = s - k, Z_w = l] \\ &= \sum_{l=1}^{T_{pop}} \mathbf{Pr}[X_{w+1} = k | X_w = l] \mathbf{Pr}[S_w = s - k, X_w = l]. \end{aligned}$$

□

Une fois cette loi conjointe déterminée, pour avoir la loi marginale, il suffit de faire la somme sur tous les états de Z_w :

$$\mathbf{Pr}[S_w = s] = \sum_{k=1}^{T_{pop}} \mathbf{Pr}[S_w = s, Z_w = k].$$

A ce stade, il est possible de vérifier la pertinence de notre hypothèse de stationnarité de $\{S_w\}$. En effet, nous connaissons l'état initial de S_1 puisqu'au temps initial, $Z_1 = 1$ (il n'y a pas d'antécédent, on ne peut pas avoir gardé le même optimum local). Donc, au début de l'algorithme, $S_1 = 1$. Or, si on calcule la loi stationnaire de S avec les paramètres que nous détaillerons dans l'application, on trouve $\mathbf{Pr}[S_1 = 1] = 0.9264$. On peut donc considérer que notre hypothèse de stationnarité n'est pas aberrante.
Nous avons déjà évoqué que la loi de $\{S_w\}$ était construite sous l'hypothèse de non convergence de l'algorithme ($\mathbf{Pr}[T_n = 0] > 0$), donc, quand l'AG va converger, son comportement va s'éloigner du comportement que nous avons modélisé et donc, on va se retrouver dans les queues de distribution de $\{S_w\}$. En fait, on s'attend à ce que, quand l'AG converge, la valeur de S_w augmente singulièrement par rapport au comportement prévu par la loi déterminée. Dans un tel cas, la probabilité pour que S_w prenne la valeur observée va devenir très faible. Il s'agit donc d'observer cette probabilité pour savoir si l'on doit s'arrêter ou non. Rappelons à nouveau qu'il s'agit d'un critère d'arrêt qui reflète une pseudo-convergence et qui n'a donc aucune garantie de correspondre à l'optimum global que, de toute façon, on ignorera toujours

(sauf dans le cas des simulations). Cependant, il permettra au moins de repérer une solution qui est de grande qualité et en faisant tourner l'AG plusieurs fois on aura une idée du *degré d'optimalité* de cette solution.
Voyons maintenant l'effet des différents paramètres inhérents au critère et les résultats sur notre simulation.

2.4 Etude du critère

2.4.1 Influence des paramètres

Etant donnée la construction du critère, nous voyons qu'il dépend d'une part des paramètres de l'AG (taille de la population, taux de mutation et de croisement, pression de sélection) et d'autre part de paramètres qui lui sont propres :

- la taille de la fenêtre, w,
- la probabilité $\mathbf{Pr}[T_n = 0]$,
- la probabilité $\mathbf{Pr}[S_w = s_{obs}]$ à partir de laquelle on considère que la valeur de S_w, s_{obs} que l'on observe est assez faible pour que l'on puisse décider que l'on est dans une situation de pseudo-convergence et donc pour laquelle on arrête l'algorithme.

C'est donc l'influence de ces trois paramètres que nous allons maintenant étudier. Notons ici, que cette étude peut se faire indépendamment du problème d'optimisation sous-jacent puisqu'il n'intervient pas dans le calcul des lois. Pour les autres paramètres, caractéristiques de l'AG, on considérera dans les paragraphes suivants que $T_{pop} = 50$, $p_m = 0.7$, $p_c = 0.5$ et que la pression de sélection est telle que le meilleur individu a une probabilité de sélection deux fois supérieure à celle de l'individu de rang médian.

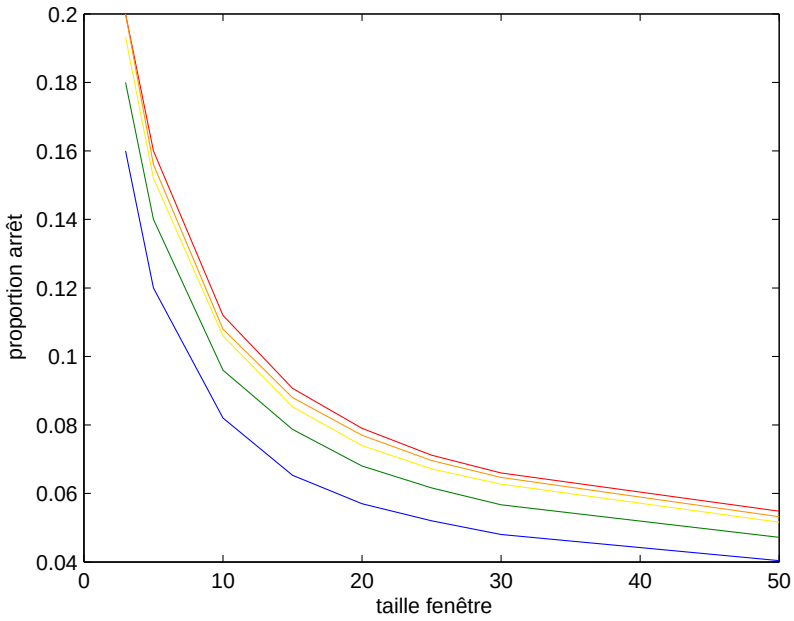
Nous allons débuter avec le paramètre $\mathbf{Pr}[T_n = 0]$. Rappelons que ce paramètre correspond à la probabilité pour l'AG de trouver une meilleure solution par rapport à la solution localement optimale. Puisque nous nous plaçons dans un contexte de non-convergence (pour pouvoir repérer la convergence comme une situation exceptionnelle), il est nécessaire que $\mathbf{Pr}[T_n = 1] < 1$. De plus, on s'attend à trouver très rapidement, au cours des générations, des solutions très intéressantes. On suppose donc que, très rapidement, la probabilité d'améliorer l'optimum local devient très faible. Nous allons étudier cinq valeurs :

$$\mathbf{Pr}[T_n = 0] \in \{0.5, 0.25, 0.1, 0.05, 0.01\}$$

et nous verrons en même temps l'évolution en fonction de la taille de la fenêtre. Les résultats sont présentés dans la Fig. 2.3.

Le critère d'arrêt représenté dans la figure correspond à la valeur de S_w à partir de laquelle on considère qu'on est en situation de pseudo-convergence et on arrête l'AG. Cette valeur est rapportée à la valeur maximale du critère, c'est-à-dire wT_{pop}. On observe ainsi une valeur indépendante de la valeur de la fenêtre qui correspond grossièrement à la proportion de la population qui doit être *colonisée* par la meilleure solution locale pour qu'on considère que l'on a atteint une pseudo-convergence. On constate que moins on a de probabilité de trouver une meilleure solution, plus cette proportion doit être grande, ce qui est tout à fait logique. Cependant, entre $\mathbf{Pr}[T_n = 0] = 0.5$ et $\mathbf{Pr}[T_n = 0] = 0.01$, il y a un écart maximal (pour $w = 3$), de 4%, soit une différence de 6 pour S_3 si $T_{pop} = 50$, ce qui n'est pas très

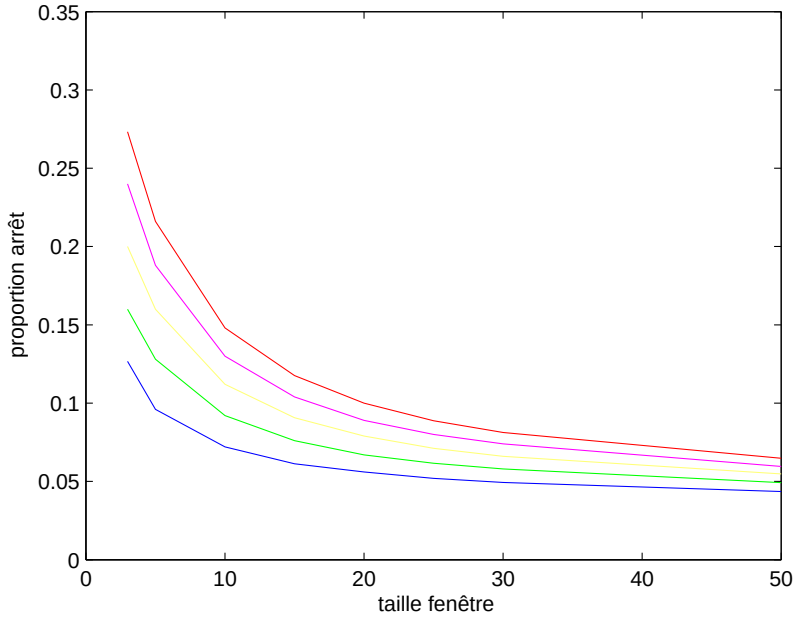
Fig. 2.3 Evolution du critère d'arrêt en fonction de la taille de la fenêtre et de $\mathbf{Pr}[T_n = 0]$. La courbe noire correspond à $\mathbf{Pr}[T_n = 0] = 0.5$, la bleue à $\mathbf{Pr}[T_n = 0] = 0.25$, la jaune à $\mathbf{Pr}[T_n = 0] = 0.1$, la rouge à $\mathbf{Pr}[T_n = 0] = 0.05$ et la courbe verte à $\mathbf{Pr}[T_n = 0] = 0.01$.



important. Cependant, afin de rendre le critère plus *sévère*, c'est-à-dire qu'il faut que le comportement de $\{S_w\}$ s'écarte vraiment du comportement modélisé, on préfèrera des valeurs de $\mathbf{Pr}[T_n = 0]$ plutôt faibles. Pour la suite, nous avons choisi $\mathbf{Pr}[T_n = 0] = 0.01$.
Concernant la taille de la fenêtre, on observe dans la Fig. 2.3 des valeurs entre 3 et 50. On s'aperçoit que pour des petites largeurs de fenêtre, la taille de la population qui doit être colonisée par la solution optimale est très importante et qu'elle diminue très rapidement. A partir d'une fenêtre de vingt générations, la diminution ralentit, c'est pourquoi nous avons choisi dans nos applications d'étudier le critère sur vingt générations.

Etudions enfin le dernier paramètre, le seuil $\mathbf{Pr}[S_w = s_{obs}]$ à partir duquel on considère que l'on est dans une situation de pseudo-convergence et qui détermine la valeur de S_w pour laquelle on s'arrête. Dans cette étude, nous gardons les mêmes paramètres caractéristiques de l'AG et nous choisissons, comme indiqué dans le paragraphe précédent, $\mathbf{Pr}[T_n = 0] = 0.01$. Les résultats sont présentés dans la Fig. 2.4. Comme dans le cas précédent, l'écart entre les différentes courbes tend à se réduire quand la taille de la fenêtre d'observation augmente. De même, la décroissance des courbes diminue quand la fenêtre augmente et particulièrement à partir d'une taille de fenêtre égale à vingt, ce qui confirme le choix indiqué précédemment. On remarque aussi que, bien que les écarts entre les seuils soient les mêmes entre courbes successives, l'écart entre les courbes a tendance à se réduire quand $\mathbf{Pr}[S_w = s_{obs}]$ diminue.

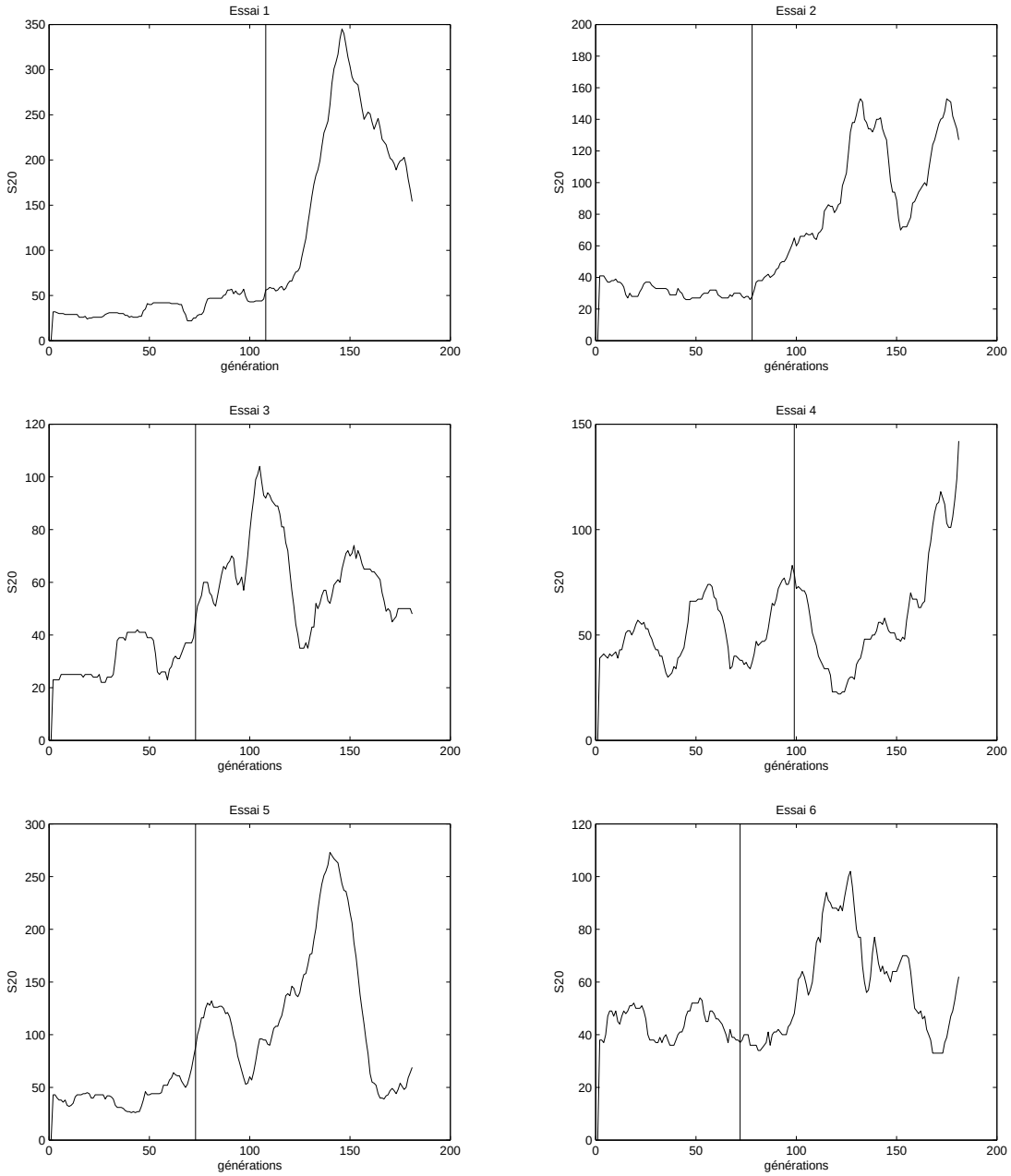
Fig. 2.4 Evolution du critère d'arrêt en fonction de la taille de la fenêtre et de la probabilité $\mathbf{Pr}[S_w = s_{obs}]$ pour laquelle on arrête l'AG. La courbe bleue correspond à $\mathbf{Pr}[S_w = s_{obs}] = 10^{-3}$, la verte à $\mathbf{Pr}[S_w = s_{obs}] = 10^{-4}$, la jaune à $\mathbf{Pr}[S_w = s_{obs}] = 10^{-5}$, la courbe rose à $\mathbf{Pr}[S_w = s_{obs}] = 10^{-6}$ et la courbe rouge à $\mathbf{Pr}[S_w = s_{obs}] = 10^{-7}$.



2.4.2 Fixation d'un seuil par une expérience

Ces observations ne permettent pas réellement de se donner un seuil. Pour étudier plus précisément ce paramètre, nous allons nous baser sur la simulation décrite dans le paragraphe 2.2.2. Nous avons appliqué l'AG utilisant l'approche décrite au paragraphe 2.2.1.1 plusieurs fois successivement au jeu de données simulé et nous avons observé l'évolution de S_{20}. Les résultats obtenus sont présentés dans la Fig. 2.5. Si on arrête l'AG quand $\mathbf{Pr}[S_w = s_{obs}] \leq 10^{-3}$ (cela correspond à $s_{obs} = 56$ pour $T_{pop} = 50$), on s'arrête trop tôt dans la moitié des cas, dans le cas d'un arrêt pour $\mathbf{Pr}[S_w = s_{obs}] \leq 10^{-4}$ ($s_{obs} = 67$), on manque l'optimum global dans deux des six essais. A partir de $\mathbf{Pr}[S_w = s_{obs}] \leq 10^{-5}$, on ne fait plus d'erreur pour ces six essais. Nous avons donc choisi, dans un premier temps, de faire plus d'essais pour $\mathbf{Pr}[S_w = s_{obs}] \leq 10^{-5}$. Nous avons donc réalisé 100 essais et nous avons obtenu le *vrai* optimum global dans 91 cas. Ceci est très satisfaisant, d'autant que pour les cas où l'optimum n'est pas atteint, un seul des individus est mal affecté pour huit des neuf cas concernés. On a donc obtenu un optimum très proche de l'optimum global. On pourrait tout à fait, pour éviter ces erreurs, choisir un seuil plus petit mais cela impliquerait forcément un temps de pseudo-convergence plus long pour un gain, en terme d'efficacité, assez restreint. Nous avons donc choisi de nous arrêter à $\mathbf{Pr}[S_w = s_{obs}] \leq 10^{-5}$.

Fig. 2.5 Evolution de S_{20} pour six applications de l'AG du paragraphe 2.2.1.1 aux données simulées. Le trait vertical indique la génération pour laquelle l'optimum global est apparu pour la première fois dans chaque essai.



2.4.3 Application à d'autres AG

Une fois les différents paramètres choisis, nous avons appliqué le critère obtenu à l'AG correspondant à la deuxième approche (paragraphe 2.2.1.2). L'intérêt de cette deuxième approche est qu'elle utilise un mode d'optimisation tout à fait différent. En fait, la convergence est beaucoup plus rapide avec cette méthode et nous allons donc voir si notre critère d'arrêt s'adapte à ce changement.
En fait, si on répète l'algorithme cent fois comme précédemment, on retrouve la solution optimale dans 100% des cas. Et si on étudie le nombre de générations nécessaires à la convergence, alors qu'en moyenne, il fallait plus de mille générations dans l'approche 1, il n'en faut plus que 30, ce qui est très proche du minimum autorisé car l'algorithme ne peut s'arrêter en moins de générations que la taille de la fenêtre (c'est-à-dire 20 générations). Le critère s'adapte donc à différentes vitesses de convergence.

Pour tester plus profondément l'adaptativité de notre nouveau critère, nous l'avons également appliqué à un AG complètement différent, celui que nous avons présenté au paragraphe 1.5 qui concerne la recherche de pics discriminants en SELDI-TOF. Dans ce cas, le calcul de la fitness est complètement différent de l'AG précédent et son évolution dans la population ne peut donc pas être la même. Afin de pouvoir connaître l'efficacité de notre critère avec certitude, nous avons simulé un jeu de données.
Rappelons que, dans cette application, les données que l'on entre dans l'AG correspondent à une matrice qui contient les différents pics en colonnes et les individus (les spectres) en ligne. Nous avons choisi de simuler cent pics et 60 individus. Les individus sont répartis en deux groupes de même effectif. Les données sont telles que l'on peut atteindre un pourcentage de bien classés de 100% en utilisant trois pics. Les autres pics ne contiennent que du bruit. L'objectif est de donc de retrouver en sortie de l'AG les trois pics intéressants avec des seuils adaptés. Comme pour les essais précédents, nous avons appliqué l'AG cent fois et étudier la solution finale obtenue au moment de l'arrêt par le critère. Finalement, 91% des essais ont mené à la bonne solution. Encore une fois, on pourrait améliorer ce résultat en prenant une probabilité d'arrêt plus faible mais la pseudo-convergence serait bien plus longue. De plus, les 9% qui n'ont pas donné la solution optimale en étaient tout de même très proches en terme de fitness. Le critère est donc également efficace pour un autre AG.

2.4.4 Limites

Nous avons vu, d'après l'hypothèse 2.2 que nous négligions la probabilité d'apparition de nouvelles solutions optimales, de même valeur que la solution optimale courante, par application des opérateurs de mutation et de croisement, devant la probabilité de disparition, par ces mêmes opérateurs, des solutions optimales déjà existantes. Dans les AG que nous venons d'étudier, cette hypothèse tient mais il existe des cas pour lesquels, il est impossible de faire cette hypothèse.

2.4.4.1 Cas 1 : peu de possibilités ou beaucoup de solutions équivalentes

Dans un cas où l'espace des solutions serait de petite dimension, par mutation et/ou croisement, on aurait une forte probabilité de retrouver la solution localement optimale à partir d'une autre solution. Cependant, on utilise généralement les AG pour des cas où l'espace des

solutions est de très grande dimension et ce cas doit donc pouvoir être évité.
Cependant, on observerait le même phénomène si de nombreuses solutions différentes obtenaient la même fitness. En effet, nous avons vu que nous comptons le nombre de solutions qui ont la valeur localement optimale, peu importe que ce soit exactement les mêmes ou qu'elles soient simplement équivalentes. Dans un tel cas, on augmenterait également la probabilité de faire apparaître de nouvelles solutions de fitness égale à la fitness optimale locale. Par exemple, on peut se trouver dans cette situation si la fitness correspond à un pourcentage de bien classés et que l'on a peu d'individus à classer. Par exemple, s'il y a vingt individus, la fitness ne pourra prendre que vingt valeurs quelle que soit la dimension de l'espace des solutions. Dans un tel cas, la modélisation de Z_n ne sera pas réaliste et le critère d'arrêt ne pourra pas être appliqué. Notons que nous avons toujours construit des fitness qui contenaient au moins deux termes, ce qui réduit les risques d'avoir beaucoup de solutions équivalentes.

2.4.4.2 Cas 2 : initialisation non aléatoire

Nous avons vu au paragraphe 1.2 de la partie 1 que l'initialisation d'un AG peut se faire selon deux grands types de méthodes : l'initialisation aléatoire et l'initialisation dirigée. Dans le cas de l'initialisation aléatoire, il n'y a pas de problème pour le critère. En revanche, si l'initialisation est dirigée et que les solutions de la population initiale sont toutes identiques ou presque, la probabilité d'obtenir plusieurs fois la même solution de façon indépendante à partir de deux solutions identiques est évidemment bien plus élevée que si toutes les solutions de départ sont différentes ou presque. Rappelons que l'on utilise généralement les AG dans le cas d'espaces des solutions de grande dimension et que l'on a donc peu de chance de générer aléatoirement plusieurs fois la même solution dans la population initiale. Le cas d'une initialisation dirigée ne se prête donc pas à l'utilisation du critère que nous avons construit.

Pour revenir aux cas auxquels nous avons appliqué le critère, pour ce qui est de la classification non supervisée, dans l'approche 1, on avait, pour chaque solution 80 loci qui pouvaient prendre 10 valeurs différentes, la dimension de l'espace des solutions est donc très importante. Quant à la fitness, elle porte sur le nombre de groupes, qui est certes limité, mais aussi sur la somme des distances intra-groupes moyennes qui, elle, peut prendre de nombreuses valeurs. Enfin, l'initialisation est aléatoire. Cette application remplit donc bien les conditions d'application du critère.
Quant à la recherche de pics discriminants en SELDI, nous avons vu au paragraphe 1.2 que la taille de l'espace des solutions est colossale. Pour la fitness, nous combinons le pourcentage de bien classés (qui prend autant de valeurs différentes qu'il y a de spectres) et le nombre de pics utilisés, ce qui représente un nombre de possibilités assez important. Enfin, l'initialisation est à nouveau aléatoire. On a donc, une fois de plus, réuni les conditions nécessaires à une bonne utilisation du critère d'arrêt que nous avons construit.

2.4.5 Conclusion

Les exemples précédents nous ont permis de montrer que le critère

$$S_n < s_{th},$$

où s_{th} est le seuil déterminé, est très efficace. Il est évident que les hypothèses simplificatrices que nous avons dû faire ne permettent pas de l'appliquer à tous les AG. De plus, nous avons

eu besoin d'une application pour pouvoir déterminer le seuil. Cependant, les développements théoriques que nous avons montrés permettent de mieux comprendre le fonctionnement de ce critère même s'ils ne recouvrent pas exactement la réalité.
C'est précisément cet écart entre le modèle et la réalité qui nous conduit à devoir considérer une probabilité très faible pour l'arrêt de notre algorithme. En effet, si on considère l'hypothèse 2.2, il est évident que, dans certains cas, de nouvelles solutions dont la fitness est égale à celle de l'optimum local peuvent apparaître parmi les autres solutions. C'est notamment le cas quand un optimum avait commencé à *coloniser* la population et qu'il a été dépassé par une autre solution. Dans une telle situation, la même solution est présente en plusieurs exemplaires et certaines peuvent évoluer dans le même sens pour donner la même solution ou des solutions de même valeur. C'est pourquoi nous avons tendance, par le modèle, à sous-estimer le nombre d'occurrences de la meilleure solution courante. Par conséquent, si on s'arrêtait à une probabilité plus élevée (0.01 par exemple), le modèle aurait l'impression que l'on a déjà atteint une colonisation importante. Pourtant, cette colonisation est due à l'apparition d'autres occurrences par mutation/sélection, il faut donc atteindre une plus grande valeur pour S_n pour que cela corresponde à l'envahissement prédit par le modèle.
De même, pour avoir une vraie estimation de $\mathbf{Pr}[T_n = 0]$, il faudrait avoir des informations précises sur la fonction que l'on optimise et tenir compte de l'évolution de l'optimum. Or, la plupart du temps, dans les applications, on n'a aucune idée des propriétés de la fonction optimisée. Tout critère qui prendrait en compte les caractéristiques de cette fonction serait donc inutilisable. Mais, on perd évidemment de l'information qui pourrait être intéressante.

Finalement, on voit bien que les hypothèses faites sont simplificatrices et qu'elles ne permettent pas de tenir compte de tous les éléments de l'AG. Mais, elles permettent d'avoir une idée assez précise de ce fonctionnement et l'utilisation de plusieurs applications nous a permis de finaliser le critère et de montrer son adaptabilité et sa efficacité, à l'intérieur des limites que nous avons citées. Il semble donc que le critère que nous avons construit réalise un bon compromis entre fondement théorique et applicabilité.

Chapitre 3

De l'accélération de la convergence

3.1 Présentation

La proposition faite dans ce paragraphe découle de l'observation suivante. Dans le cas où un optimal local s'est *installé* pendant plusieurs générations et que son nombre d'occurrences dans la population devient important, si un nouvel optimum apparaît, il aura du mal à s'imposer dans la population. En effet, surtout près de la convergence, l'ancien optimum a beaucoup de chances d'avoir une valeur juste inférieure à celle du nouveau. Il aura donc des rangs très proches, ce qui, en termes de probabilité de sélection, se traduit par une très légère différence. D'autant plus que la probabilité globale de sélection de l'optimum précédent est multipliée par le nombre d'occurrences de cette solution. C'est pourquoi le nombre d'occurrences du nouvel optimum aura du mal à augmenter.
Ce phénomène peut avoir d'importantes conséquences sur la convergence de l'algorithme, notamment dans le cadre de l'application du critère de convergence construit dans le paragraphe précédent. Pour proposer une solution à ce problème, nous sommes retournés à la biologie afin d'essayer de voir s'il existait une analogie à ce phénomène et si la nature avait *trouvé* une solution.
A nouveau, il est possible de trouver une similitude dans l'histoire de l'évolution des espèces. En effet, on sait qu'au-delà des phénomènes de mutation/croisement/sélection, d'autres phénomènes ont des conséquences sur les apparitions/disparitions d'espèces. Parmi ceux-ci, on trouve les grandes catastrophes. Pour exemple, nous prendrons la plus connue d'entre toutes, celle qui marque la limite entre l'ère secondaire et l'ère tertiaire, celle qui a vu la disparition des dinosaures. A la fin du secondaire, les dinosaures étaient des animaux dominants sur terre qui limitaient le développement d'autres groupes, comme les mammifères. Une fois que les dinosaures se sont éteints (par l'une ou l'autre des multiples causes actuellement proposées...) les mammifères (et beaucoup d'autres groupes) ont eu le *champ libre* pour se développer. Certaines qualités leur ont permis de résister à la crise, ils étaient donc, en cela, mieux adaptés que les dinosaures et la crise a permis de faire s'exprimer cet avantage. Notons que cette théorie a connu de très récentes remises en question (Bininda-Emonds *et al.*, 2007), mais nous laisserons de côté ces polémiques qui dépassent le cadre de notre étude.
Revenons à nos AG, nous nous trouvons exactement dans le même cas : des individus très bien adaptés (au sens de la fitness) ne peuvent exprimer leur avantage en raison de la suprématie d'autres individus bien adaptés. Pour résoudre le problème, nous allons donc simuler un épisode d'extinction. Plus précisément, nous allons, de temps en temps, laisser la possibilité de détruire les individus dont le nombre d'occurrences est le plus grand et les remplacer par

Tab. 3.1 Nombre moyen de générations nécessaires à la convergence pour l'application à l'AG pour les k-means dans le cas de l'AG simple et de l'AG avec catastrophe pour $k = 1$, 2 et 3.

	sans cata	$k = 2$	$k = 3$	$k = 4$
nb générations	135	113	121	105

les individus localement optimaux.
Ce phénomène a évidemment des conséquences sur le déroulement de l'AG, c'est pourquoi, à chaque fois que l'on appliquera la catastrophe, on remettra à zéro les compteurs de la convergence. Pour cela, on réinitialise la variable aléatoire S_w à sa valeur minimum, c'est-à-dire w (la taille de la fenêtre). On ignore donc ce qui s'est passé avant la catastrophe mais ce n'est pas gênant puisqu'on sait que la dépendance entre les valeurs successives de Z_n ne porte que sur l'étape précédente.

3.2 Application

Nous avons appliqué cette idée à l'algorithme introduit dans le paragraphe 2.2.1.2, celui qui réalise une sorte de k-means où l'AG gère le nombre et la position des centres de groupes. Nous avons repris le même type simulation que précédemment et appliqué notre algorithme cent fois de suite. A chaque itération, le jeu de données est simulé à nouveau, on trouve donc des jeux de données légèrement différents d'une étape à l'autre. Pour chaque jeu de données simulé, on applique l'AG décrit dans le paragraphe 2.2.1.2 et le même algorithme auquel on ajoute des catastrophes aléatoires. Pour gérer la fréquence des ces catastrophes, nous avons généré un nombre entre 0 et 1 et si ce nombre était inférieur à $1/(k \times w)$. Il semble en effet que w, la taille de la fenêtre est une bonne base pour définir la fréquence des catastrophes. En effet, il ne semble pas raisonnable de prévoir des catastrophes qui seraient plus fréquentes que la taille de la fenêtre. Nous avons donc, pour chaque simulation, fait tourner l'algorithme avec catastrophe pour $k \in \{2, 3, 4\}$. Nous avons enregistré le nombre de générations nécessaires à la convergence dans chaque cas, les résultats sont donnés dans la Tab. 3.1.
Les résultats obtenus nous permettent difficilement de faire un choix quant à la fréquence nécessaire. On peut tout de même remarquer que c'est pour $k = 4$ que l'on obtient, en moyenne, le plus petit nombre de générations nécessaires à la convergence. En tous cas, on voit nettement une diminution du nombre de générations nécessaires lorsqu'on introduit les événements de catastrophe. Il semble donc que ce processus soit tout à fait favorable à l'accélération de la convergence.
Finalement, on constate que les phénomènes naturels peuvent apporter certaines solutions. Il peut donc être profitable, lorsqu'on rencontre un problème au niveau de l'algorithme, de chercher s'il n'existe pas de problème analogue dans la nature et comment l'obstacle a été franchi. Il est bien évident que cette technique ne peut pas résoudre tous les problèmes mais elle peut apporter certaines idées fructueuses.

Conclusion

Nous allons tout d'abord dresser un bilan des conclusions qui ont pu apparaître au cours de ce travail.

Dans la première partie, nous avons présenté une sorte d'*état de l'art* des AG. Son objectif était de permettre une bonne compréhension de ces méthodes et non de cette méthode car nous avons vu qu'il n'existait pas un mais bien plusieurs AG. C'est cette richesse et ses conséquences que nous avons voulu exposer dans cette partie.

Dans le premier chapitre, nous avons considéré la mise en place d'un AG et nous avons vu que l'on peut distinguer cinq grandes étapes : le codage, l'initialisation, la mutation, le croisement et la sélection. Nous pouvons déduire de cette présentation qu'il existe de très nombreuses possibilités, pour la moindre étape. Malheureusement, on possède rarement un critère qui pourrait nous dire, *à coup sûr*, quel choix effectuer, quel que soit le contexte d'application. Il est donc nécessaire d'avoir une connaissance d'un bon nombre de possibilités de sorte à choisir, pour chaque cas, la ou les solutions les mieux adaptées. Suivant l'application, on privilégiera la rapidité, la précision ou l'adaptabilité. En tous cas, il est très souhaitable de faire les choix en connaissance de cause.
Nous l'avons vu, rien n'a permis (et ne permettra sans doute jamais), de prouver une hypothétique supériorité des AG sur d'autres méthodes d'optimisation. Ce qui leur apporte le succès, c'est essentiellement le fait que l'on puisse les adapter pour tenir compte, au mieux, des contraintes de l'application. Ces remarques peuvent rebuter certains mais l'effort est réellement payant. De plus, il existe certains principes assez simples. Par exemple, nous avons choisi de réaliser un codage réel (et non binaire) car cela nous semble plus adapté pour des problèmes optimisant de nombreux paramètres. On obtient des chromosomes plus courts et cela permet d'interpréter chaque étape, notamment les mutations.

Nous avons ensuite évoqué les résultats existants sur la modélisation des AG et leur utilisation pour obtenir des résultats de convergence. Bien que de tels efforts aient été faits dès l'introduction des AG, il semble que c'est l'introduction d'une modélisation grâce aux chaînes de Markov qui ait permis un grand pas en avant à la compréhension des AG. Ces méthodes ont l'avantage d'être tout à fait adaptées au fonctionnement des AG mais également celui de *traîner* dans leur sillage de très nombreux résultats théoriques. On peut les appliquer aux AG plus ou moins directement afin d'en déduire certaines propriétés théoriques.
Ces efforts de modélisation ont permis une bien meilleure compréhension des mécanismes qui sous-tendent les AG. Cela pourrait ainsi permettre leur diffusion plus large et faire, peu à peu, disparaître les *a priori* négatifs qui considèrent les AG comme des boîtes noires mystérieuses. Par ailleurs, si les résultats obtenus permettent de comprendre les raisons des succès des AG,

ils expliquent également certains échecs (comme les problèmes de convergence) et permettent souvent d'y remédier. Cependant, bien que ces modélisations aient permis de franchir l'étape nécessaire qui consistait à *théoriser* les AG, les conséquences pratiques de ces efforts ne sont pas aussi importantes que l'on aurait pu s'y attendre. Ceci est particulièrement vrai pour les résultats de convergence.

Le dernier chapitre de cette première partie nous a permis d'introduire les principaux représentants de la famille des métaheuristiques. Une famille haute en couleur dans laquelle on voit passer des *fourmis*, des *recuits*, des *mutants*,... Pour chacune des méthodes présentées, il existe de nombreuses variantes, comme nous l'avons montré pour les AG au début de cette partie. Ce que nous avons présenté est une sorte de socle commun qui rassemble ces variantes. Nous avons essayé, dans la mesure du possible, de faire une présentation unifiée de ces méthodes de sorte à mettre en évidence leurs points communs.
Mais notre apport principal dans ce domaine est la proposition d'une nomenclature pour permettre de réaliser une comparaison assez précise des divers algorithmes. L'objectif n'en est absolument pas de faire ressortir telle ou telle méthode comme supérieure ou, au contraire, inefficace. Comme pour le premier chapitre, son objectif est de donner des outils de connaissance pour permettre un éventuel choix plus éclairé. En effet, les critères que nous avons retenus nous semblent constituer les principaux éléments qui décrivent le mode de fonctionnement des algorithmes et donc les plus importants pour connaître leur applicabilité à chaque problème d'optimisation.
Nous n'avons pas (ou peu) appliqué les autres méthodes proposées, mais notre expérience des AG nous laisse penser qu'une certaine expertise est également nécessaire pour réaliser une application efficace de telle ou telle méthode. Cependant, étant donnés les nombreux points communs entre algorithmes, il nous semble qu'une fois que l'on maîtrise une des méthodes la *prise en mains* des autres algorithmes doit être beaucoup plus rapide. Rappelons ici qu'il n'existe pas de méthode qui soit meilleure que les autres de façon universelle. Les succès rencontrés par telle ou telle méthode viennent généralement d'une bonne maîtrise de l'outil et d'un réel effort d'adaptation au problème d'optimisation posé.

Dans la deuxième partie, nous nous sommes essentiellement intéressés à l'application concrète des AG à deux types de données provenant du domaine de la protéomique, discipline dont nous espérons avoir convaincu notre lecteur de l'utilité. Cette partie avait deux objectifs, le but premier était de proposer aux biologistes des outils performants et adaptés à leurs données qui leur apportent une réelle valeur par rapport aux méthodes existantes. Ceci est passé par la mise en place de méthodes statistiques originales totalement indépendantes de la mise en œuvre des AG. Dans un deuxième temps, cette partie avait pour objectif de montrer au lecteur la nécessité d'une adaptation minutieuse de chaque étape de l'AG à chaque problématique.

La première application concernait les données de spectrométrie de masse SELDI. L'objectif était de trouver des protéines (correspondant à des pics à l'intérieur des spectres) qui soient caractéristiques de telle ou telle condition biologique. Il s'agissait donc d'un choix de variables dans le cadre d'un problème de discrimination. Pour cette application, nous avons mis au point une méthode de traitement du signal (extraction des pics et alignement des spectres).
Cependant, l'essentiel de notre travail a porté sur la conception d'une méthode de discri-

mination. Cette méthode a été conçue de sorte à être mieux adaptée aux caractéristiques des données. Les conséquences de la très grande variabilité des données d'intensité ont été réduites par une binarisation des données. Pour cela, nous avons défini un seuil d'intensité qui était optimisé à l'intérieur de l'AG. La taille généralement réduite des échantillons nous a conduits à utiliser les informations de différents pics simultanément et non séquentiellement comme dans les arbres de classification. Enfin, comme les expériences biologiques confrontent souvent plus de deux conditions, nous avons utilisé la technique du *pairwise coupling* pour étendre notre méthode de discrimination à plus de deux groupes.
La méthode a été appliquée à deux jeux de données. Le premier est un *classique* de la littérature, certes controversé, mais qui nous a permis de comparer nos résultats avec ceux de nombreuses autres méthodes. Les pics que nous avons sélectionnés comme discriminants avaient déjà été repérés par les autres méthodes mais la combinaison que nous avons obtenue était tout à fait originale et efficace. La méthode a également été appliquée à des données originales, les résultats ont beaucoup intéressé les biologistes qui isolent actuellement les protéines correspondantes pour les étudier plus profondément.
Dans le cadre de la spectrométrie et du choix de variables, nous avons également mis au point un AG pour la recherche de longueur d'ondes discriminantes pour des données de spectrométrie Proche Infra-Rouge. La demande provenait d'un industriel. Les résultats ont été validés en laboratoire et publiés.

La deuxième application concernait la deuxième grande technique utilisée en protéomique : l'électrophorèse en deux dimensions. Le problème consistait à aligner les spots de différents gels. Là encore, il a été nécessaire de tenir compte des différentes caractéristiques des données. Tout d'abord, il était connu qu'une transformation globale n'était pas adéquate, il fallait donc réaliser plusieurs transformations locales. Nous avons choisi d'utiliser des landmarks qui nous ont permis d'avoir quelques points de repères ainsi qu'une grille (sous forme de cellules de Voronoï) pour diviser les gels en cellules élémentaires. Puisque l'on utilise des landmarks, la forme et la position des cellules sont adaptées à chaque gel (ce qui n'est pas le cas quand on divise le gel en rectangles, comme cela est fait habituellement). Un alignement global a été réalisé et les résultats obtenus, notamment l'erreur commise après alignement, ont servi à définir des zones homogènes dans lesquelles une transformation locale a été définie.
Nous avons tenu à mettre au point une méthode qui ne se base pas sur un gel de référence. En effet, la définition d'un gel de référence (qu'il s'agisse d'un des gels choisi plus ou moins au hasard ou d'un gel construit) empêche l'appariement de spots qui ne seraient pas présents dans le gel de référence mais dans plusieurs autres gels. Etant donné que l'on cherche généralement à aligner des gels provenant de différentes conditions biologiques, on s'attend à avoir de nombreuses apparitions/disparitions de spots dont on risque de manquer une partie avec un gel de référence. Nous avons donc conçu une extension de la méthode Procuste généralisée qui permet d'aligner plusieurs tableaux de points dont on ne connaît pas l'appariement à l'avance. Les résultats obtenus sur un jeu de données réel a permis de constater que, bien que l'on n'ait imposé aucune contrainte, les différentes transformations locales sont tout à fait cohérentes entre elles. On constate une continuité d'une zone à l'autre ce qui montre la pertinence du découpage et des transformations trouvées. Les résultats des appariements ont été validés par le biologiste qui était à l'origine des données et sont beaucoup plus complets que les résultats obtenus avec les logiciels habituellement utilisés.

Dans la dernière partie, nous nous sommes intéressés à la convergence des AG. Cette

propriété est fondamentale tant du point de vue pratique que théorique. C'est pourquoi nous avons cherché à obtenir des résultats sur ces deux plans. En effet, les résultats théoriques sont généralement très complets et précis mais les informations nécessaires à leur utilisation pratique ne sont généralement pas disponibles. Cependant, ces résultats sont indispensables pour justifier l'usage de telles méthodes. En ce qui concerne la pratique, les utilisateurs se contentent généralement d'utiliser des critères empiriques tels qu'un nombre maximum de générations ou une évolution non significative de la fitness moyenne dans les populations. Malheureusement, ces critères doivent être réévalués pour chaque nouvelle application. C'est pourquoi nous avons cherché à concilier ces deux points de vue.

Le premier chapitre était consacré à l'extension des résultats de convergence pour les AG avec élitisme obtenus par Bhandari *et al.* (1996). Ce chapitre a permis de montrer comment on pouvait faire entrer les AG dans le cadre des chaînes de Markov et comment on pouvait s'en servir. En effet, la modélisation directe des populations des AG est extrêmement complexe, le nombre de populations est, le plus souvent, colossal, conduisant à un espace d'état de très grande dimension, difficile à gérer. Nous avons évoqué dans cette partie une façon de réduire la dimension de l'espace des solutions en regroupant les états qui, du point de vue de la convergence, sont équivalents. Pour ce faire, les populations sont tout d'abord réduites à la meilleure valeur de fitness que l'on trouve parmi leurs individus puis regroupées par valeurs de fitness équivalentes.
Notre apport dans ce domaine a consisté à étendre les résultats au cas d'une chaîne de Markov non homogène et à des mutations dirigées. Pour ce qui concerne la non homogénéité, l'objectif était de pouvoir inclure dans les résultats un taux de mutation variable qui peut permettre de contrôler la convergence. On autorise une large exploration de l'espace des solutions au début de l'algorithme puis une diminution du taux de mutation permet à la population de converger. On peut aussi envisager une réaugmentation du taux de mutation à la fin de l'AG pour lui permettre de *sortir* d'un éventuel optimum.
Nous avons également introduit l'idée de mutation dirigée qui consiste à pouvoir choisir un type de mutation par individu au cours d'une étape. Dans les résultats de Bhandari *et al.* (1996), chaque locus avait la même probabilité de mutation ce qui permettait de s'assurer que l'on puisse passer de n'importe quelle solution à n'importe quelle autre *en n'importe quel nombre donné* de générations. Cependant, on peut vouloir diriger les mutations en définissant des grands types de mutation et en appliquant une seule mutation pour un même individu au cours d'une étape. Par exemple, dans le cadre du SELDI, dans un comité, nous autorisons l'algorithme à ajouter un pic, à supprimer un pic ou à modifier un seuil, de façon exclusive. Cela permet d'introduire certaines priorités dans les mutations obtenues. En revanche, il est impossible, dans ce cas, de pouvoir obtenir n'importe quelle solution à partir de n'importe quelle autre *en n'importe quel nombre donné* de générations. Nous avons étendu la démonstration à notre cas en montrant qu'il suffisait de pouvoir obtenir n'importe quelle solution à n'importe quelle autre *en un nombre fini* de générations.
Pour remplir cette condition, les contraintes sont beaucoup plus lâches : il suffit d'avoir défini son opérateur de mutation de sorte à ce que tous les changements existants soient possibles, ce qui semble être une propriété minimale à imposer pour construire un opérateur de mutation. Nous avons démontré que la convergence était toujours valide pour ces cas, ce qui permet d'envisager des AG plus souples et donc plus à même d'être adaptés précisément à différentes problématiques.

Le chapitre suivant a eu pour but la construction d'un critère de convergence aux bases théoriques fortes mais dont l'application ne devrait pas nécessiter des informations généralement indisponibles. Pour cela, nous avons recouru au nombre d'occurrences de l'optimum local au cours des générations. En effet, un *bon* optimum est censé *coloniser* la population. Nous avons donc modélisé le nombre d'occurrences de la meilleure solution courante par une chaîne de Markov qui tient uniquement compte des paramètres de l'AG et de son comportement quand il n'a pas encore convergé. Le critère ne nécessite pas d'information sur la fonction que l'on est en train d'optimiser ce qui rend son utilisation indépendante du problème d'optimisation. La modélisation nous a permis de donner les formules de calcul des probabilités de transition de cette chaîne. A partir de ces résultats, et à l'aide d'une formule de récurrence que nous avons démontrée, nous avons pu déterminer la loi d'une somme de réalisations successives de cette chaîne. C'est finalement cette loi que nous avons utilisée pour proposer un critère d'arrêt de l'algorithme.
Le critère ne nécessite la détermination que de trois paramètres : le nombre de générations successives que l'on considère, la probabilité de non convergence et la probabilité pour laquelle on souhaite arrêter l'AG. Ces paramètres ont l'avantage d'être immédiatement interprétables, on sait exactement à quoi ils correspondent. Nous avons montré dans le détail comment ces paramètres ont été fixés et quelle est leur influence sur les résultats de convergence. Le prix à payer est évidemment que l'on ne peut être sûr que l'optimum final est global. En revanche, on a la garantie d'obtenir une solution très intéressante et plusieurs applications de l'AG permettent de tester l'adaptativité de cet optimum.
Nous avons également défini les limites de l'application de ce critère. Ces limites imposent certaines contraintes quant à l'initialisation de l'AG et écartent les applications pour lesquels on a trop de solutions équivalentes ou pour lesquelles la fitness ne peut prendre qu'un petit nombre de valeurs différentes. Nous avons choisi plusieurs applications qui se situaient évidemment à l'intérieur de ces limites et l'application de ce critère a donné d'excellents résultats.

Enfin, nous avons voulu donner une solution possible à certains problèmes de convergence qui sont liés à la colonisation de la population par un optimum local qui empêche, par la suite, la colonisation par un nouvel optimum plus intéressant. Cette solution est, une nouvelle fois, inspirée par les phénomènes naturels de l'évolution. Il s'agit de simuler les effets d'une catastrophe comme celle qui a vu la disparition des dinosaures au profit d'autres groupes comme les mammifères. De plus, nous ne voulions pas que ce procédé soit incompatible avec l'utilisation du critère de convergence que nous avons proposé. La variable aléatoire, S_w définie précédemment a donc été réinitialisée à chaque fois que l'on applique une catastrophe. D'après les résultats que nous avons obtenus, on note une accélération non négligeable de la convergence.

Nous espérons avoir montré, au cours de ce travail, les différents aspects des méthodes riches et puissantes que sont les AG. Des méthodes aux origines tout à fait intuitives mais aux fondements théoriques maintenant établis. Même si cette démarche (intuition puis démonstration) est certainement la plus courante et la plus fructueuse pour la mise en place de méthodes efficaces, il nous semble assez fabuleux d'arriver à modéliser un phénomène naturel (aux concepts très simples, même si les mécanismes en jeu sont extrêmement sophistiqués) par une méthode aux développements théoriques si complexes (chaînes de Markov, théorie de Freidlin-Wentzell) avec le succès que l'on a montré. Si on pense en outre, qu'après

être passés par les raisonnements plus ou moins abstraits des statisticiens, les AG finissent par venir en aide aux biologiques, on peut considérer que *la boucle est bouclée*.

Perspectives

Pour ce qui est des limites à l'utilisation des AG, il nous semble qu'il en est une qui peut être dépassée rapidement et qui ne devrait pas rentrer en ligne de compte, c'est le temps de calcul. En effet, toutes les méthodes que nous avons mises au point ont été programmées en langage R ou Matlab. Les fonctions écrites sont diffusables mais, pour ne pas alourdir le manuscrit, nous ne les avons pas jointes. On pourrait certainement diminuer considérablement les temps de calculs en utilisant des langages comme C ou Fortran. De même, l'utilisation d'ordinateurs plus puissants que ceux que nous utilisons permettrait un réel gain de temps. Nous pensons donc que le temps de calcul ne devrait pas être un critère de rejet des AG d'autant qu'ils ne sont pas toujours très longs. De plus, les applications pour lesquelles nous les avons utilisés peuvent se permettre de prendre du temps. L'alignement des gels par exemple prendra toujours moins de temps que les jours de travail qu'il faut au biologiste pour les aligner *à la main*.
Toujours dans un souci de meilleure diffusion des AG il serait certainement nécessaire d'implémenter les méthodes conçues dans des logiciels facilement utilisables par des non-spécialistes. En effet, en les englobant dans des *boîtes à outils*, une fois que la problématique a bien été étudiée, il est tout à fait possible, pour les biologistes, d'appliquer ces méthodes à de nouveaux jeux de données en ne jouant que sur un nombre minimal de paramètres. Des interfaces graphiques et des explications concrètes de chacun des paramètres utilisés devraient permettre une telle utilisation.
En ce qui concerne les perspectives plus statistiques, il nous semble qu'il reste encore de nombreuses possibilités d'exploration, notamment, pour permettre la mise en place de critères de convergence réellement utilisables. Cette notion de critère de convergence nous semble primordiale, c'est sans doute l'aspect le plus limitant actuellement pour la diffusion des AG. Nous avons essayé d'y contribuer mais nous avons également montré les limites du critère que nous avons construit. Ces limites sont probablement difficiles à dépasser dans le cadre d'un critère qui se veut indépendant du problème d'optimisation. Mais, il est certainement possible de poursuivre dans ce sens, notamment en prenant en compte au fur et à mesure de l'AG les informations que l'on acquiert au cours des générations sur la fitness. En effet, il n'est généralement pas envisageable d'avoir une information précise *a priori* concernant les données. Cependant, au fur et à mesure des générations, on peut certainement se servir de ce qui se passe dans les populations pour *apprendre* des informations sur la fitness. Cela nous permettrait de réaliser une modélisation plus précise qui s'adapterait à chaque problème.
Enfin, le cas de l'ajout efficace de la catastrophe, présenté dans le dernier chapitre, peut également laisser entrevoir d'autres possibilités dans l'insertion dans les AG de phénomènes qui ont fait leurs preuves dans la nature. Bien sûr de tels ajouts nécessiteraient, à leur tour, des études théoriques, mais nous avons vu combien cet effort peut être payant. On pourrait penser, par exemple, à construire des sous-populations qui évolueraient indépendamment avant de se rejoindre. Cela permettrait sans doute d'augmenter la probabilité d'atteindre l'optimum global.
Bref, même si les AG sont des méthodes que l'on pourrait considérer comme *vieillottes* (elles datent tout de même de 1975), nous pensons qu'il existe encore beaucoup de chemins prometteurs à explorer, tant d'un point de vue théorique que pour les domaines d'application.

Nous sommes certains qu'un tel progrès ne peut passer que par une étroite collaboration entre *théoriciens* et *appliqués* et par une large diffusion des résultats obtenus dans les deux domaines.

Bibliographie

Aarts, E.H.L. & van Laarhoven, P.J.M. (1985) Statistical cooling : A general Approach to combinatorial optimization problems. *Phillips Journal of Research*, **40**, 193-226.

Aarts, E.H.L. & Korst, J. (1989) *Simulated Annealing and Boltzmann Machines : A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, Chichester, England.

Alexe, G., Alexe, S., Liotta, L.A., Petricoin, E., Reiss, M. & Hammer, P.L. (2004) Ovarian cancer detection by logical analysis of proteomic data. *Proteomics*, **4**, 766-783.

Antonisse, J. (1989) A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint. *Proceedings of the Third International Conference on GAs*, Morgan Kaufmann, Publishers.

Baggerly, K.A., Morris, J.S., Wang, J., Gold D., Xiao, L.-C. & Coombes, K.R. (2003) A comprehensive approach to the analysis of matrix-assisted laser desorption/ionization-time of flight proteomics spectra from serum samples. *Proteomics*, **3**(9), 1667-1672.

Baker, J.E. (1987) Reducing bias and inefficiency in the selection algorithm. In : Grefenstette, J.J. (Ed.) *Proceedings of the 2nd International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, New jersey, 14-21.

Barker, M. & Rayens, W. (2003) Partial Least Squares for discrimination. *Journal of Chemometrics*, **17**,166-173.

Bauchart, C., Remond, D., Chambon, C., Patureau Mirand, P., Savary-Auzeloux, I., Reynès, C. & Morzel M. (2006) Small peptides (¡5 kDa) found in ready-to-eat beef meat. *Meat Science*, **74**, 658-666.

Bhandari, D., Murthy, C.A. & Pal, S.K. (1996) Genetic algorithm with elitist model and its convergence. *International Journal of Pattern Recognition and Artificial Intelligence*, **10**(6), 731-747.

Bininda-Emonds, O.R.P., Cardillo, M., Jones, K.E., MacPhee, R.D.E., Beck, R.M.D., Grenyer, R., Price, S.A., Vos, R.A., Gittleman, J.L. & Purvis, A. (2007) The delayed rise of present-day mammals. *Nature*, **446**, 507-512.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984) *Classification and regression trees*, Wadsworth/Brooks/Cole, Monterey.

Bremermann, H.J. (1962) *Optimization through Evolution and Recombination*, Spartan Books, San Francisco, CA.

Cerf, R. (1994) *Une théorie asymptotique des algorithmes génétiques*, Thèse de doctorat, Université Montpellier II.

Cerf, R. (1996a) The dynamics of mutation-selection algorithms with large population sizes. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, **32**(4), 455-508.

Cerf, R. (1996b) A new genetic algorithm. *Annals of Applied Statistics*, **6**(3), 778-817.

Ciphergen ProteinChip® Software, Version 3.1 (2002), Ciphergen Biosystems, Freemont, CA, USA.

Commandeur, J.J.F. (1991) *Matching configurations*, Leiden, The Netherlands : DSWO Press.

Coombes, K.R., Fritsche, H.A., Clarke, C., Chen, J.-N., Baggerly, K.A., Morris, J.S., Xiao, L.-C., Hung, M.-C. & Kuerer, H.M. (2003) Quality Control and Peak Finding for Proteomics Data Collected from Nipple Aspirate Fluid by Surface-Enhanced Laser Desorption and Ionization. *Clinical Chemistry*, **49**(10), 1615-1623.

Darwin, C.R. (1859) *On the Origin of Species by means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*, John Murray, London.

Davenport, A., Tsang, E.P.K., Wang, C.J. & Zhu, K. (1994) GENET : a connectionist architecture for solving constraint satisfaction problems by iterative improvement. *Proceedings 12th National Conference for Artificial Intelligence (AAAI)*, 325-330.

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

Davis, T.E. & Principe, J.C. (1993) A Markov chain framework for the simple genetic algorithm. *Evolutionary Computation*, **1**, 269-288.

Dawkins, R. (1995) *River out of Eden*, Basic Books, New York.

De Jong, K.A. (1975) *An analysis of the behaviour of a class of genetic adaptive systems*, Doctoral Dissertation, University of Michigan, Ann Arbor, Michigna.

De Jong, K.A. (1992) Are genetic algorithms function optimizers ? In : Männer, R. & Manderick, B. (Eds.) *Parallel Problem-Solving from Nature, 2*, Elsevier Science Publishers, Amsterdam, 3-13.

Dijksterhuis, G. & Gower, J.C. (1991) The interpretation of generalized Procrustes analysis and allied methods. *Food Quality and Preference*, **3**, 67-87.

DiMatteo, I., Genovese, C.R. & Kass, R.E. (2001) Bayesian curve-fitting with free-knot splines. *Biometrika Trust*, **88**, 1055-1071.

Dorigo, M. & Di Caro, G. (1999) The Ant Colony Optimization meta-heuristic. In : Corne, D., Dorigo, M. & Glover, F. (eds.), *New Ideas in Optimization*. McGraw Hill, London, 11-32.

Dorigo, M., Di Caro, G. & Gambardella, L.M. (1999) Ant algorithms for discrete optimization. *Artificial Life*, **5**(2), 137-172.

Duda, R., Hart, P. & Strok, D. (2001) *Pattern classification and scene analysis*, John Wiley and Sons, New York.

Eshelman, L.J., Caruana, R.A. & Schaffer, J.D. (1989) The CHC adaptive search algorithm : How to hava safe search when engaging in nontraditional genetic recombination. In : Rawlins, G.J.E. (Ed.) *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 265-283.

Faigle, U. & Kern, W. (1992) Some convergence results for probabilistic Tabu Search. *ORSA Journal on Computing*, **4**, 32-37.

Feo, T.A. & Resende, M.G.C. (1989) A probabilisstic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, **29**, 330-341.

Feo, T.A. & Resende, M.G.C. (1995) Greedy Randomized Adaptive Search procedures. *Journal of Global Optimization*, **6**, 109-133.

Figuères G., Andrieu E., Letourneux S., Biesse J.P & B. Vidal (2004) *Near infrared spectroscopy and pattern recognition as screening methods for classification of commercial tobacco blends*. Congrès de Chimiométrie, Paris, France, 30 Novembre & 1er Décembre 2004.

Fogarty, T.C. (1989) Varying the probability of mutation in the genetic algorithm. In : Schaffer, J.D. (Ed.) *Proceedings of the third Internation Conference on Genetic Algorithms*, Morgan Kaufmann, Los ALtos, CA, 104-109.

Fogel, L.J. (1963) *Biotechnology : Concepts and Applications*, Prentice-Halls, Englewood Cliffs, NJ.

Fogel, L.J., Owens, A.J. & Walsh, M.J. (1966) *Artifical Intelligence through Simulated Evolution*, John Wiley, New York.

Fogel, D.B. (1998) *Evolutionary Computation : The Fossil Record*, IEEE Press, Piscataway, NJ.

Fortune, S. (1987) A Sweepline Algorithm for Voronoi Diagrams. *Algorithmica*, **2**, 153-174.

Fraser, A.S. (1962) Simulation of genetic systems, *Journal of Theoretical Biology*, **2**, 329-346.

Freidlin, M.I. & Wentzell, A.D. (1984) *Random perturbations of dynamical systems*, Springer-Verlag, New York.

Friedman, J. H. (1984) *A variable span scatterplot smoother*. Laboratory for Computational Statistics, Stanford University Technical Report No. 5.

Gendreau, M. (2003) An introduction to tabu search. In : Glover, F. & Kochenberger, G.A. (eds.), *Handbook of metaheuristics*, Kluwer's International Series, 37-54.

Gidas, B. (1985) Non stationary markov chains and convergence of the annealing algorithm. *Journal of Statistical Physics*, **39**, 73-131.

Glover, F. (1986) Future paths for integer programming and links to artificial intelligence. *Computers and operations research*, **13**, 533-549.

Glover, F. (1998) A template for scatter search and path relinking. In : Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M. & Snyers, D. (eds.), *Artificial evolution, Lecture Notes in Computer Science* 1363, Springer, 3-51 Klose.

Glover, F. & Hanafi, S. (2002) Tabu Search and Finite Convergence. *Discrete Applied Mathematics*, **119**, 3-36.

Glover, F., Laguna, M. & Marti, R. (2003) Scatter search and path relinking : advances and applications. In : Glover, F. & Kochenberger, G.A. (eds.), *Handbook of metaheuristics*, Kluwer's International Series, 1-35.

Goldberg, D.E. (1985) *Optimal initial population size for binary-coded genetic algorithms*, TCGA Report 85001, University of Alabama, Tuscaloosa.

Goldberg, D.E. (1989a) Sizing populations for seria and parallel genetic algorithms. In : Schaffer, J.D. (Ed.) *Proceedings of the third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 70-79.

Goldberg, D.E. (1989b) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachussetts.

Goldberg, D.E., Deb, K. & Clark, J.H. (1992) Genetic algorithms, noise and the sizing of populations. *Complex Systems*, **6**, 333-362.

Gower, J.C. (1975) Generalized Procrustes Analysis. *Psychometrika*, **40**, 33-51.

Green, B.F. (1952) The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, **17**, 429-440.

Green, P.J. (1995) Reversible jump Markov Chain Monte Carlo computation and bayesian model determination. *Biometrika*, **82**, 711-732.

Gustafsson, J.S., Blomberg, A. & Rudemo, M. (2002) Warping two-dimensional electrophoresis gel images to correct for geometric distorsions of the spot pattern. *Electrophoresis*, **23**, 1731-1744.

Gutjahr, W.J. (2002) ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, **82**, 145-153.

Hanafi, S. (2001) On the convergence of Tabu Search. *Journal of Heuristics*, **7**(1), 47-58.

Hancock, P.J.B. (1994) An empirical comparison of selection methods in evolutionary algorithms. In : Fogarty, T.C. (Ed.) *Evolutionary Computing : AISB Workshop, Leeds, UK, April 1994 ; Selected Papers*, Springer-Verlag, Berlin, 80-94.

Hancock, P.J.B. (1996) Selection methods for evolutionary algorithms. In : Chambers, L. (Ed.) *Practical Handbook of Genetic Algorithms : New Frontiers, Vol. II*, CRC Press, Boca Raton, FL, 67-92.

Hastie, T., Tibshirani, R. & Friedman, J. (2001) *The Elements of Statistical Learning, Data Mining, Inference and Prediction*, Springer-Verlag, New York.

Henderson, D., Jacobson, S.H. & Johnson, A.W. (2003) The theory and practice of simulated annealing. In : Glover, F. & Kochenberger, G.A. (eds.), *Handbook of metaheuristics*, Kluwer's International Series, 287-319.

Hilario, M., Kalousis, A., Müller, M. & Pellegrini, C. (2003) Machine learning approaches to lung cancer prediction from mass spectra. *Proteomics*, **3**(9), 1716-1719.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan.

Hurley, J.R. & Cattell, R.B. (1962) The Procrustes Program : Producing direct rotation to test a hypothesized factor structure. *Behavioural Science*, **7**, 258-262.

James, P. (1997) Protein identification in the post-genome era : the rapid rise of proteomics. *Quarterly review of biophysics*, **30**, 279-331.

Janikow, C.Z. & Michalewicz, Z. (1991) An experimental comparison of binary and floating point representations in genetic algorithms. In : Belew, R.K. & Booker, J.B. (Eds.), *Proceedings of the Fourth International Conference Genetic Algorithms*, Morgan Kaufmann, San Mateo, 31-36.

Jeffers, J.N.R. (1967) Two case studies in the application of principal component analysis. *Applied Statistics*, **16**, 225-236.

Jeffries, N.O. (2004) Performance of a genetic algorithm for mass spectrometry proteomics. *BMC Bioinformatics*, **5**(180).

Jong, K., Marchiori, E. & van der Vaart, A. (2004) Analysis of proteomic pattern data for cancer detection. *Lecture Notes in Computer Science*, **3005**, 41-51.

Kaczmarek, K., Walczak, B., de Jong, S. & Vandeginste, B.G. (2004) Preprocessing of two-dimensional gel electrophoresis images. *Proteomics*, **4**, 2377-2389.

Klose, J. (1975) Protein mapping by combined isoelectric focusing and electrophoresis : a two-dimensional technique. *Humangenetik*, **26**, 231-234.

Kristof, W. (1970) A theorem on the trace of certain matrix products and some applications. *Journal of Mathematical Psychology*, **7**, 515-530.

Kristof, W. & Wingersky, B. (1971) Generalization of the orthogonal Procrustes rotation procedure for more than two matrices. In : *Proceedings of the 79th annual convention of the American psychological association*, p.89-90.

Le, R., Hino, M. & Ezzo, M. (2005) How antiquated statistics can overcome showy youth. *Journal of Adequate Statistic Methods*, **18**(5).

Leardi, R. (2000) Application of genetic algorithm-PLS for feature selection in spectral data sets. *Journal of Chemometrics*, **14**, 643-655.

Lee, K.R., Lin, X., Park, D.C. & Eslava, S. (2003) Megavariate data analysis of mass spectrometric proteomics data using latent variable projection method. *Proteomics*, **3**(9), 1680-1686.

Liu, H., Li, J. & Wong, L. (2002) A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Informatics*, **13**, 51-60.

Lohr, S.L. (1999) *Sampling : Design and Analysis*, Duxbury Press, Pacific Grove, CA.

Lundy, M. & Mees, A. (1986) Convergence of an annealing algorithm. *Mathematical Programming*, **34**, 111-124.

Mardia, K.V., Kent, J.T., Bibby, J.M. (1979), *Multivariate Analysis*, Academic Press, London.

Marengo, E., Robotti, E., Gianotti, V., Righetti, P.G., Cecconi, D., Domenici, E. (2003) A new integrated statistical approach to the diagnostic use of two-dimensional maps. *Electrophoresis*, **24**, 225-236.

McKay, M.D., Conover, W.J. & Beckman, R.J. (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**, 239-245.

Michalewicz, Z. (1992) *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, Berlin.

Mitra, D., Romeo, F. & Sangiovanni-Vincentelli, A. (1986) Convergence and finite time behavior of simulated annealing. *Advances in Applied Probability*, **18**, 747-771.

Mladenovic, N. (1995) A Variable neighborhood algorithm - a new metaheuristic for combinatorial optimization. In : *Abstracts of papers presented at Optimization Days*, Montréal, p.112.

Mladenovic, N. & Hansen, P. (1997) Variable neighborhood search. *Computers Operations Research*, **24**, 1097-1100.

Monod, J. (1970) *Le Hasard et la Nécessité*. Le seuil, Paris, p.155.

Mülenbein, H. & Schlierkamp-Voosen, D. (1994) The science of breeding and its application to the breeder genetic algorithm. *Evolutionary computation*, **1**, 335-360.

Muri-Majoube, F. & Prum, B. (2002) Une approche statistique de l'analyse des génomes. *La Gazette des Mathématiciens*, **89**, 63-98.

Naes, T. & Mevik, B.-H. (2001) Understanding the collinearity problem in regression and discriminant analysis. *Journal of Chemometrics*, **15**,413-426.

Nix, A.E. & Vose, M.D. (1992) Modelling genetic algorithms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, **5**, 79-88.

O'Farrell, P.H. (1975) High resolution two-dimensional electrophoresis of proteins. *Journal of Biological Chemistry*, **250**, 4007-4021.

Peck, C.C. & Dhawan, A.P. (1995) Genetic algorithms as global random search methods : An alternative perspective. *Evolutionary Computation*, **3**, 39-80.

Peng, T., Jian, M. & Zhiping, F. (1997) A stochastic tabu search strategy and its global convergence. *IEEE International conference on Computational Cybernetics and Simulations*, **1**, 410-414.

Petricoin, E.F., Ardekani, A.M., Hitt, B.A., Levine, P.J., Fusaro, V.A., Steinberg, S.M., Mills, G.B., Simone, C., Fishman, D.A., Kohn, E.C. & Liotta, L.A. (2002) Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, **359**, 572-577.

Pittman, J. (2002) Adaptive Splines and Genetic Algorithms. *Journal of Computational & Graphical Statistics*, **11**(3), 615-638.

Potra, F.A. & Liu, X. (2006) Aligning Families of 2D-gels by a Combined Multiresolution Forward-Inverse Transformation Approach. *Journal of Computational Biology*, **13**(7), 1384-1395.

Prados, J., Kalousis, A., Sanchez, J.-C., Allard, L., Carrette, O. & Hilario, M. (2004) Mining mass spectra for diagnosis and biomarker discovery of cerebral accidents. *Proteomics*, **4**(8), 2320-2332.

Qu, Y., Adam, B.-L., Yasui, Y., Ward, M.D., Xazares, L.H., Schellhammer, P.F., Feng, Z., Semmes, O.J. & Wright, G.L. (2002) Boosted Decision Tree Analysis of Surface-enhanced Laser Desorption/Ionization Mass Spectral Serum Profiles Discriminates Prostate Cancer from Noncancer Patients. *Clinical Chemistry*, **48**(10), 1835-1843.

R Development Core Team (2004). *R : A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http ://www.R-project.org.

Radcliffe, N.J. & Surry, P.D. (1995) For mae and the variance of fitness. In : Whitley, D. & Vose, M. (Eds.), *Foundations of the Genetic Algorithms 3*, Morgan Kaufmann, San Mateo, CA, 51-72.

Rechenberg, I. (1973) *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipen der biologischen Evolution*, Frommann-Holzboog Verlag, Stuttgart. (2nd Edition 1993).

Rees, J. & Koehler, G.J. (1999) An investigation of GA performance results for different cardinality alphabets. In : Davis, L.D., De Jong, K.A., Vose, M.D. & Whitley, L.D. (Eds.) *Evolutionary algorithms :* IMA Volumes in Mathematics and its Applications, **111**, Springer-Verlag, New-York, 191-206.

Reeves, C.R. (1994) Genetic algorithms and neighbourhood search. In : Fogarty, T.C. (Ed.) *Evolutionary Computing : AISB Workshop, Leeds, UK, April 1994 ;Selected Papers.* Springer-Verlag, Berlin.

Reeves, C.R. (1995) A genetic algorithm for flowshop sequencing. *Computers & Operations research*, **22**, 5-13.

Reeves, C.R. (2000) Fitness landscapes and evolutionary algorithms. In : Fonlupt, C., Hao, J.K., Lutton, E., Ronald, E. & Schoenauer, M. (Eds.) *Artificial Evolution : 4th European Conference ; Selected Papers.* Springer-Verlag, Berlin, 3-20.

Reeves, C.R. & Rowe, J.E. (2003) *Genetic algorithms - Principles and perspectives, A guide to GA theory*, Kluwer Academic Publishers, London.

Reynès, C., de Souza, S., Sabatier, R., Figuères, G., Vidal, B. (2007a) Selection of discriminant wavelength intervals in NIR spectrometry with genetic algorithms. *Journal of Chemometrics*, **20**, 136-145.

Reynès, C., Roche, S., Tiers, L., Sabatier, R., Jouin, P., Molinari, N. & Lehmann, S. (2007b) Comparison between surface and bead based MALDI profiling technologies using a single bioinformatics algorithm. *Clinical Proteomics* (sous presse).

Rogers, M., Graham, J. & Tonge, R.P. (2003) Statistical models of shape for the analysis of protein spots in two-dimensional electrophoresis gel images. *Proteomics*, **3**, 887-896.

Rousson, V. & Gasser, T. (2003), Some Case Studies of Simple Component Analysis. *Department of Biostatistics*, University of Zürich. (Available from http ://www.unizh.ch/biostat/Manuscripts/.)

Rousson, V. & Gasser, T. (2004) Simple Component Analysis. *Applied Statistics*, **53**, 539-555.

Rudolph, G. (1997) *Convergence properties of Evolutionary Algorithms*. Verlag Dr. Kovac, Hamburg, Germany.

Salmi, J., Aittokallio, T., Westerholm, J., Griese, M., Rosengren, A., Numan, T.A., Lahesmaa, R. & Nevalainen, O. (2002) Hierarchical grid transformation for image warping in the analysis of two-dimensional gel electrophoresis. *Proteomics*, **2**, 1504-1515.

Schönemann, P.H. & Carroll, R.M. (1970) Fitting one matrix to another under choice of a central dilatation and a rigid motion. *Psychometrika*, **35**, 245-256.

Schwefel, H.-P. (1977) *Numerische Optimierung von Computer-modellen mittels der Evolutionsstrategie*, Birkhäuser Verlag, Basel. (English edition : *Numerical Optimization of Computer Models*, John Wiley & Sons, Chichester, 1981).

Shapiro, J.L., Prügel-Bennett, A. & Rattray, M. (1994) A statistical mechanics formulation of the dynamics of genetic algorithms. *Lecture Notes in Computer Science.*, **865**, 17-27.

Smilansky, Z. (2001) Automatic registration for images of two-dimensional protein gels. *Electrophoresis*, **22**, 1616-1626.

Sorace, J.M. & Zhan, M. (2003) A data review and re-assessment of ovarian cancer serum proteomic profiling. *BMC Bioinformatics*, **4**(24).

Stadler, P.F. & Wagner, G.P. (1998) Algebraic theory of recombination spaces. *Evolutionary Computation*, **5**, 241-275.

Stewart, W.J. (1995) *Introduction to the numerical solution of Markov Chains*, Princeton University Press, Princeton.

Ten Berge, J.M. (1977) Orthogonal procrustes rotation for two or more matrices. *Psychometrika*, **42**, 267-276.

Tenenhaus, M. (1998) *La Régression PLS, Théorie et Pratique*, Technip, Paris, 1998.

Thomas, E.V. (1994) A primer on multivariate calibration. *Analytical Chemistry*, **66**, 795-804.

Tibshirani, R., Hastie, T., Narasimhan, B., Soltys, S., Shi, G., Koong, A. & Le, Q.-T. (2004) Sample classification from protein mass spectrometry, by peak probability contrasts. *Bioinformatics*, **20**(17), 3034-3044.

Tsang, E.P.K., Wang, C.J., Davenport, A., Voudouris, C. & Lau, T.L. (1999) A family of stochastic methods for constraint satisfaction and optimisation. *Proceedings of the First International Conference on The Practical Application of Constraint Technologies and Logic Programming (PACLP)*, London, 359-383.

van Laarhoven, P.J.M. & Aarts, E.H.L. (1987) *Simulated Annealing : Theory and Applications*. D. Reidel, Kluwer Academic Publishers, Dordrecht, Boston, Norwell, Massachusetts.

Veeser, S. Dunn, M.J. & Yang, G.Z. (2001) Multiresolution image registration for two-dimensional gel electrophoresis. *Proteomics*, **1**, 856-870.

Vines, S.K. (2000) Simple Principal Components. *Applied Statistics*, **49**, 441-451.

Vose, M.D. (1988) Generalizing the notion of schema in genetic algorithms. *Artificial Intelligence*, **50**, 385-396.

Voss, T. & Haberl, P. (2000) Observations on the reproducibility and matching efficiency of two-dimensional electrophoresis gels : consequences for comprehensive data analysis. *Electrophoresis*, **21**, 3345-3350.

Wagner, M., Naik, D. & Pothen, A. (2003) Protocols for disease classification from mass spectrometry data, *Proteomics*, **3**, 1692-1698.

Wang, X. & Feng, D.D. (2005) Hybrid registration for two-dimensional gel protein images. In : Yi-Ping Phoebe Chen & Limsson Wong (eds.), *Proceedings of 3rd Asia-Pacific Bioinformatics Conference*, 17-21 January 2005, Singapore, Imperial College Press, London.

Watson, J.D & Crick, F. (1953) A Structure for Deoxyribose Nucleic Acid. *Nature*, **171**, 737-738.

Whitney, D. (1989) The GENITOR algorithm and selection pressure : Why ranked-based allocation of reproductive trials is best. In : Schaffer, J.D. (Ed.) *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan-Kaufmann, Los Altos, CA 116-121.

Wolpert, D.H. & Macready, W.G. (1997) No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67-82.

Wu, T.-F., Lin, C.-J. & Weng, R.C. (2004) Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, **5**, 975-1005.

Yasui, Y., Pepe, M.S., Thompson, M.L., Adam, B.-L., Wright, G.L., Qu, Y., Potter, J.D., Winget, M., Thornquist, M. & Feng, Z. (2003) A data-analytic strategy for protein biomarker discovery : profiling of high-dimensional proteomic data for cancer detection. *Biostatistics*, **4**(3), 449-463.

Ycart, B. (2002) *Modèles et algorithmes markoviens*, Mathématiques et Applications, **39**, Springer, Berlin, 270p.

Zhu, W., Wang, X., Ma, Y., Rao, M., Glimm, J. & Kovach, J.S. (2003) Detection of cancer-specific markers amid massive mass spectral data. *Proceedings of the National Academy of Sciences*, **100**(25), 14666-14671.