INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités

I. C. Lerman - R. F. Ngouenet

N° 2570

Janvier

PROGRAMME 3

ISSN 0249-6399

INRIA
RENNES

Algorithmes génétiques séquentiels et parallèles pour une représentation affine des proximités

I. C. Lerman - R. F. Ngouenet

Programme 3 — Intelligence artificielle, systèmes cognitifs
et interaction homme-machine
Projet Repco

Rapport de recherche n° 2570 — Janvier — 62 pages

Résumé : Nous présentons une nouvelle approche de la représentation affine des matrices de proximité utilisant les algorithmes génétiques (AG). Les AG représentent des techniques d'optimisation stochastiques inspirées directement de la théorie darwinienne, sur les mécanismes de l'évolution naturelle des êtres vivants et de la génétique. Ainsi, nous montrons comment les opérations fondamentales des AG, "croisement" et "mutation" peuvent être interprètées de façon synthétique et adéquate dans la problématique de la représentation euclidienne.

De tels algorithmes possèdent un coût d'exécution assez important pour les matrices de grande taille. En prenant en considération certaines propriétés inhérentes aux AG, nous proposons d'une part une stratégie d'hybridation et d'autre part, un algorithme parallèle orienté architecture multi-processeur à mémoire distribuée. Une implémentation sur la machine Paragon donne un speed-up presque linéaire. Nous présentons des résultats numériques attestant de la validité de la méthode.

Mots-clé: Représentation affine; Algorithme génétique parallèle; Paragon; speed-up.

(Abstract: pto)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Serial and parallel genetic algorithms for multidimensional scaling

Abstract: In this paper, a new approach to multidimensional scaling (MDS) using genetic algorithms (GAs) and its acceleration by parallelization are presented. GAs are stochastic search and optimization techniques based on the mechanics of natural selection and natural genetics. Specifially, the fundamental basic operations of these algorithms "crossover" and "mutation" are successfully adapted to the euclidean representation in a synthetic and suitable way.

Unfortunately, serial standard GAs with large populations suffer from lack of efficiency (quite high execution time). Thus, an hybridization strategy is suggested and a massively parallel genetic algorithm for MDS is proposed. An implementation on Intel Paragon supercomputer is given and the parallel algorithm shows a near-linear speed-up. We illustrate the usefulness of the approach by providing some application examples.

Keywords: Multidimensional scaling; Parallel genetic algorithm; Intel Paragon supercomputer; speed-up.

Table des matières

1 Introduction

Ce travail est à la croisée de trois domaines: l'analyse métrique des données, les algorithmes génétiques et le calcul parallèle.

Le problème à résoudre concerne, plus précisément, la représentation affine - le plus souvent planaire - d'une matrice de proximités (dissimilarités ou similarités) $\Delta = \{\delta_{ij}/1 \leq i,j \leq n\}$ sur un ensemble fini E d'entités. L'origine ou le mode de construction d'une telle matrice ne nous concernent aucunement ici.

Les méthodes numériques couramment employées pour obtenir des solutions, difficilement évaluables, de cette problématique sont regroupées sous le terme générique MDS (MultiDimensional Scaling) [7, 1989]. Ces méthodes procèdent en optimisant des fonctions coût dont la forme analytique est difficile à gérer [37, 1992].

Toutefois, lorsque la matrice Δ est *euclidienne*, une solution analytique de ce problème - PCA[1] ou AFTD - s'obtient à partir de la décomposition spectrale de la matrice $W(\Delta)$ de Torgerson [4, 1976].

Les solutions que nous avons proposées dans [33, 1994] et que nous reprenons ici avec plus de détails, consistent à adopter, pour l'optimisation de ces fonctions coût, et ceci indépendamment des modèles de la MDS, une approche basée sur les Algorithmes Génétiques (AG).

Les AG constituent une large famille d'algorithmes stochastiques dont la première version, présentée dans la première partie et couramment appelée Algorithme Génétique Canonique (AGC), fut introduite par Holland [24, 1975], redynamisée et popularisée par Goldberg [18, 1989]. Nous présentons également dans cette partie une modélisation et une étude du comportement asymptotique de l'AGC au moyen d'une chaîne de Markov. Le résultat important qu'on démontre est que tout AGC ne converge pas nécessairement vers un optimum global quel que soit le paramètrage que l'on se fixe [cf. §2.4].

Ces algorithmes inspirés directement de la théorie darwinienne, sur l'évolution des êtres vivants, opèrent une recherche stochastique sur un espace de taille exponentielle avec trois opérations fondamentales: reproduction, croisement et mutation. Nous donnons, dans la deuxième partie, à chacune de ces opérations et au codage des individus, une expression synthétique directement liée à la sémantique de notre problème de représentation affine [cf. §3]. Ainsi, nous maîtrisons la signification géométrique de ces opérateurs tout en diminuant considérablement la taille de l'espace de recherche. Cependant, la complexité de nos algorithmes demeure importante. D'où une proposition d'hybridation de nos algorithmes par des algorithmes de des-

1. Principal Coordinate Analysis

cente [cf. §3.6]. Nous décrivons les effets du *sharing* - fonction de partage - ([20, 1987] et [17, 1989]) dans la convergence de nos algorithmes [cf. §3.7]. Une approche récente [36, 1992] a consisté à construire des configurations initiales avec l'algorithme stochastique de recuit simulé.

Une grande part des calculs que suppose un algorithme génétique étant par nature parallélisable, nous proposons dans la troisième partie des versions parallèles de nos algorithmes pour les calculateurs multiprocesseurs à mémoire distribuée. C'est dans cette partie que nous prenons connaissance des outils et des termes du parallélisme.

La quatrième partie regroupe des exemples démonstratifs sur des données réelles. Nous terminons par une conclusion évoquant de nouvelles perspectives.

2 Les Algorithmes génétiques

2.1 Préambule

Les Algorithmes Génétiques (AG) représentent une famille assez riche et très intéressante d'algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Les champs d'application sont fort diversifiés. On les retrouve aussi bien en théorie des graphes qu'en compression d'images numérisées ou encore en programmation automatique [18, 1983],[27, 1984],[8, 1990],[41, 1987]. Les raisons de ce nombre d'applications sont claires. Leur prinicipe est d'opérer une recherche stochastique sur un important espace à travers un ensemble - une population - de pseudo-solutions. Ces algorithmes sont simples et très performants dans leur recherche d'amélioration. De plus, ils ne sont pas limités par des hypothèses contraignantes sur le domaine d'exploration. Ainsi, le mathématicien abordant le sujet n'a guère à se préoccuper de la continuité et de la différentiabilité de la fonction à optimiser. Une autre technique d'exploration stochastique qui a actuellement beaucoup de succès, le *recuit simulé*[2], utilise un processus aléatoire pour aider à guider la recherche détats d'énergie minimale. Elle est fondée sur une analogie avec le recuit physique des solides, qui consiste à chauffer un matériau à haute température, et à le faire refroidir très lentement afin de laisser le système atteindre son énergie minimale. Toutefois, les algorithmes de recuit simulé font évoluer une seule pseudo-solution. Cette caractéristique majeure fait d'ailleurs du recuit simulé une méthode fondamentalement séquentielle [43, 1990] et [1, 1989]. Un ouvrage récent [11, 1987] explore les relations entre le recuit simulé et les algorithmes génétiques. Une re-

2. Un algorithme de recuit simulé est décrit en page 8

marque importante qui en sort est qu'une exploration pseudo-aléatoire n'implique pas nécessairement une exploration sans direction.

En resumé, les algorithmes génétiques diffèrent fondamentalement des autres méthodes selon quatre axes principaux:

1. Les AG utilisent un codage des paramètres, et non les paramètres eux mêmes.
2. Les AG travaillent sur une population de points, au lieu d'un point unique.
3. Les AG n'utilisent que les valeurs de la fonction étudiée, pas sa dérivée, ou une autre connaissance auxiliaire.
4. Les AG utilisent des règles de transition probabilistes, et non déterministes.

Afin de bien appréhender la dynamique des algorithmes génétiques, introduisons les principaux éléments du jargon utilisé dans la littérature.

V1 *Individu* ou *Chromosome :* une solution potentielle;

V2 *Population :* un ensemble de chromosomes ou de points de l'espace de recherche;

V3 *Environnement :* l'espace de recherche;

V4 *Fitness* ou *Fonction d'évaluation :* la fonction - positive - que l'on cherche à maximiser.

Ce petit lexique, qui fait montre d'une prudente analogie avec la biologie, a naturellement une simple fonction métaphorique. Leur emploi n'obeit qu'à un souci pragmatique. Nous les utilisons dans l'introduction de l'algorithme génétique standard, considéré desormais par maintes auteurs sous le nom d' Algorithme Génétique Canonique (AGC), défini par Holland [24, 1975] et redynamisé par Goldberg [17, 1989].

Dans ce paragraphe nous développons de façon précise les principaux aspects de la théorie des Algorithmes génétiques. Nous commencerons par exprimer au paragraphe 2.2 en quoi consiste un AGC. Nous y décrivons les opérateurs fondamentaux des AG qui sont le croisement et la mutation. Au paragraphe 2.2.1 nous présentons le théorème fondamental des AG. Nous terminerons ce paragraphe par une analyse de la convergence des AG fondée sur la théorie markovienne des processus stochastiques dans un espace d'états fini.

Algorithme: **recuit simulé**

Début
$i \leftarrow 0$
T température initiale
X_c solution initiale
Evaluer X_c
Répéter
Répéter
Générer une autre solution X_n
dans le voisinage de X_c
Si $f(X_c) < f(X_n)$
alors $X_c \leftarrow X_n$
Sinon
$\tilde{p}$ réel tiré dans $(0,1)$
$p = exp\{(f(X_n) - f(X_c))/T\}$
Si $\tilde{p} < p$ alors $X_c \leftarrow X_n$
fsinon
fsi
jusqu'à condition posée.
$T \leftarrow g(T,i) < T$ diminuer la température.
$i \leftarrow i + 1$
jusqu'à arret.
Fin

2.2 Algorithme génétique canonique

Avant d'examiner les mécanismes et la puissance d'un Algorithme Génétique Canonique (AGC), presentons clairement l'objectif visé. Le problème de l'AGC peut être résumé de la manière suivante: on dispose d'une fonction f, non constante, définie sur le cube logique $\{0,1\}^N$ - N entier naturel - à valeurs dans $\mathbb{R}_+^*$ et on se fixe pour objectif de localiser l'ensemble f^* des maxima globaux de f, ou à défaut, de trouver le plus rapidement et le plus efficacement possible des points sous-optimaux. Formellement, il s'agit d'une optimisation sans contrainte du type:

$$max\{f(x)/x \in \{0,1\}^N\}$$

Très souvent, comme dans notre problème de la MDS, on s'intéresse plutôt aux minima d'une fonction f; et, dans ce contexte, les résultats suivants sont utilisés.

▷ **1 (Optimisation des fonctions)** *Soit f une fonction réelle a une ou plusieurs variables. Les problèmes d'optimisation suivants sont équivalents:*

(1) $min_x f(x)$

(2) $max_x\{-f(x)\}$

De plus, la fonction optimisée par un AG - f - doit avoir des valeurs positives dans l'espace de recherche. Dans le cas contraire, il est nécessaire d'ajouter aux valeurs de f une constante positive adéquate C conformément à l'équivalence des problèmes d'optimisation suivants:

▷ **2** **(a)** $max_x f(x)$

(b) $max_x\{C + f(x)\},\ C\ \in \mathbb{R}$

En définitive, l'objectif de l'AG est de régler les différents paramètres d'un problème donné pour obtenir la plus grande valeur du fitness f possible.

2.2.1 Principe de l'AGC

L'organigramme [cf. figure 1] nous montre succinctement le fonctionnement de l'AGC et indique clairement ses différents opérateurs de base. Ces opérateurs sont inspirés directement des mécanismes de la sélection naturelle et des phénomènes génétiques. Ainsi, le principe de l'AGC - ce qui d'ailleurs justifie son nom - est de faire évoluer une population de façon à adapter les individus à l'environnement en place, comme le ferait un ensemble d'êtres vivants. Ce principe s'apparente convenablement à celui de la théorie de Darwin sur l'évolution, selon laquelle " la vie est une compétition et seuls les mieux adaptés survivent et se reproduisent ".

Techniquement, une nouvelle génération s'obtient généralement au bout d'un cycle [cf. figure 1] composé de trois opérateurs standards [17, 1989]: **reproduction, croisement, et mutation**.

O1 **Reproduction** L'opérateur de reproduction directement inspiré de la sélection naturelle consiste à dupliquer chaque individu de la population proportionnellement à sa valeur d'adaptation. Il s'agit, en quelque sorte, de réaliser autant de

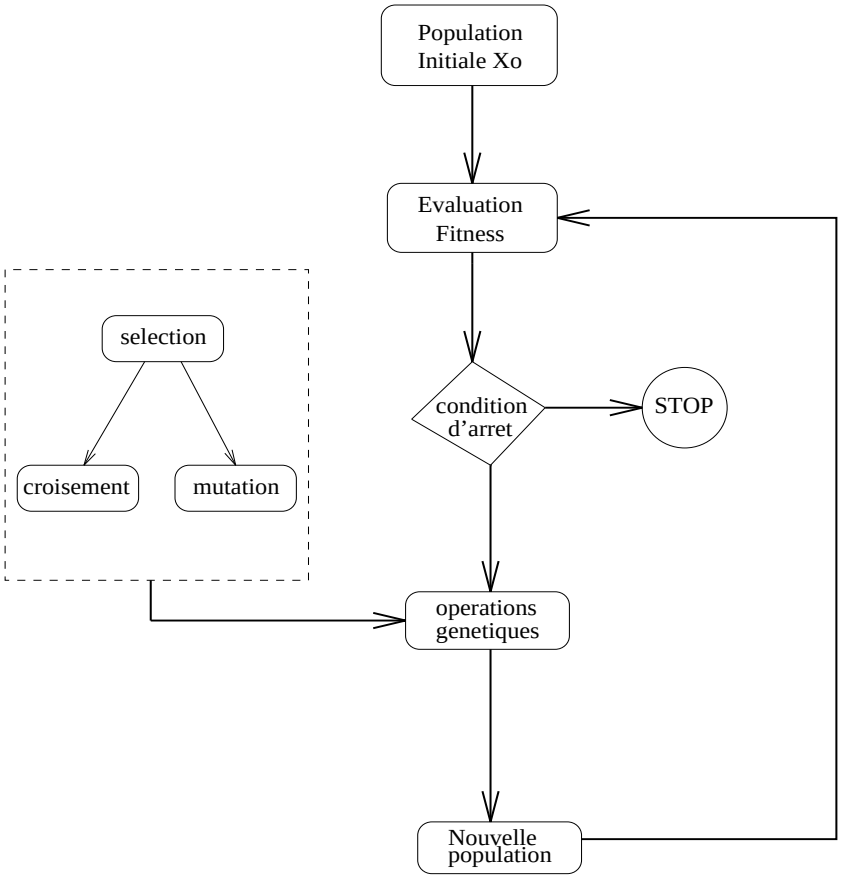


FIG. 1 - *Organigramme de l'ACG*

tirages avec remise - à la manière de la technique du Bootstrap[3] - qu'il y a d'éléments dans la population. Chaque élément étant tiré avec une probabilité liée à sa valeur d'évaluation que le statisticien peut bien interpréter comme des probabilités de survie. Ainsi, les individus ayant un fitness grand ont plus de chance d'être sélectionnés pour la génération suivante. On parle alors de sélection proportionnelle. L'opérateur de reproduction peut être mis en oeuvre sous forme algorithmique de différentes façons. La plus facile est de créer une roue de loterie biaisée [cf. figure 2

3. Bien que ce mot évoque le fait de se soulever soi-même en tirant sur ses lacets, son principe reste une idée très importante en statistique.

] pour laquelle chaque individu de la population courante occupe une section de la roue proportionnelle à son adaptation.

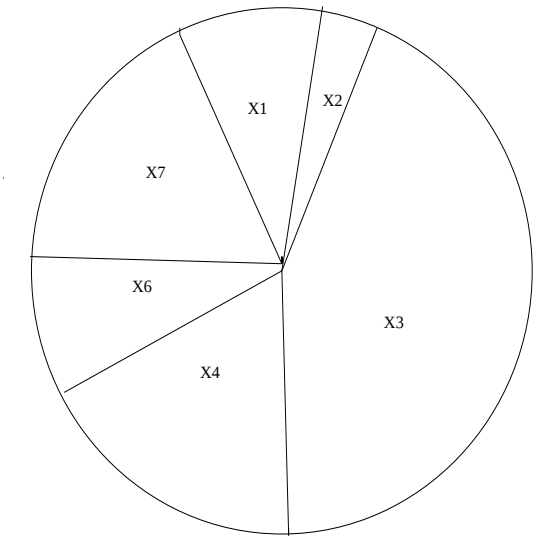


FIG. 2 - *Roue de loterie biaisée*

Pour réaliser la reproduction, il suffit de faire tourner la roue biaisée ainsi définie autant de fois qu'on a besoin de descendant. Le groupe ainsi obtenu constitue l'ébauche de la nouvelle génération qui est ensuite soumise à d'autres opérateurs génétiques.

Etant donné un fitness positif f et une taille de la population n, une telle reproduction peut être décrite schématiquement par le pseudo code suivant:

Algorithme: **roue biaisée**

Début
Evaluer le fitness f_i de chaque individu X_i
Calculer F le fitness total de la population
F = $\sum_{i=1}^{n} f_i$
Pour i:= 1 : n
$p_i = f_i/F$ probabilité de sélection de X_i
$q_i = 0$
Pour j:= 1 : i
$q_i = q_i + p_j$
fpour
q_i fixe la probabilité cumulative de X_i
fpour
Pour i:= 1 : n
$\tilde{p}$ réel tiré dans $[0, 1]$
Si $\tilde{p} < q_1$
Sélectionner X_1
Sinon
Sélectionner X_k, $2 \leq k \leq n$
tel que $q_{k-1} < \tilde{p} \leq q_k$
fsi
fpour
Fin

L'inconvénient majeur de ce type de reproduction vient du fait qu'elle peut favoriser la domination d'un individu ne représentant pas véritablement le meilleur. De plus, elle peut aussi engendrer une perte de diversité par la domination d'un super-individu. Ainsi, dans l'exemple ci-dessus [cf. figure 3] où la population courante est

$$\{X1, X2, X3, X4, X5, X6, X7\}$$

on peut malheureusement obtenir à la génération suivante, une population constituée du même individu. Par exemple

$$\{X6, X6, X6, X6, X6, X6, X6\}$$

qui n'est manifestement pas une "bonne" population. Pour pallier cet inconvénient, on préfère très souvent des méthodes de reproduction plus justes qui n'autorisent en aucun cas la naissance d'un super-individu. Par exemple, la sélection proportionnelle au rang ou d'autres techniques faisant intervenir des notions de voisinage entre chromosomes.

Après la reproduction, un croisement permet de générer les nouveaux individus.

O2 Croisement Relativement à deux individus " parents ", considérés à cette occasion comme des vecteurs, la dynamique de l'opérateur croisement, inspirée directement du processus de multiplication des chromosomes humains, consiste à générer deux nouveaux individus " enfants " de la façon suivante :

⋆ **Croisement à un site** Un site de croisement est tiré aléatoirement sur l'individu, puis les segments finaux des deux parents sont échangés autour de ce site. Illustrons cette opération dans le cas où notre espace de recherche est un espace d'arbres dont les l'étiquetage des noeuds se fait de gauche à droite. Pour cela, considérons deux graphes $G1$ et $G2$ ayant 9 noeuds [cf. figure 3].

En supposant que le site de croisement issu d'un tirage au sort est le noeud 2 (resp. 4) pour le graphe G1 (resp. G2), on obtient conformément à ($\star$) deux enfants G3 et G4 donnés par la figure 4.

⋆⋆ **Croisement uniforme** Chaque composante du nouveau vecteur est tirée aléatoirement parmi les composantes des parents ayant le même indice. Généralement le second enfant est construit en prenant le complémentaire du premier enfant comme nous l'indique le schéma 5.

Les opérations de croisement que nous venons de décrire sont les plus utilisés dans la littérature des AG. Il en existe bien d'autres tels que le croisement multi-site ou bien des croisements spécialisés liés à la sémantique du problème traité et/ou à la structure du codage considéré. Signalons déjà que cette opération de croisement prendra dans le cadre de la MDS un aspect très spécifique et original. Notons au passage que l'opérateur de croisement ne modifie pas nécessairement, compte tenu du caractère aléatoire qu'il revêt, tous les individus ayant survécu à la reproduction.

O3 Mutation L'opération de mutation consiste à modifier de manière aléatoire, avec une probabilité généralement faible, la valeur d'une composante de l'individu. Dans le cas du codage binaire de l'AGC, un booléen $w \in \{0,1\}$ choisi aléatoirement est remplacé par son inverse $1-w$. Techniquement, la combinaison de la reproduction

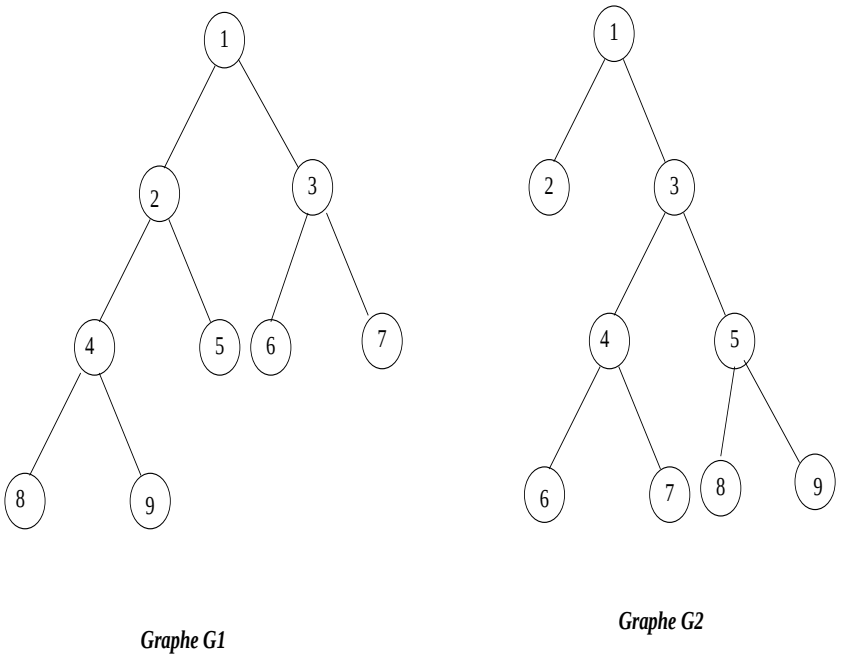


FIG. 3 - *Graphes à croiser*

et du croisement est suffisante pour assurer l'évolution de la population, mais il peut arriver que certaines informations essentielles - composantes dans la cas des vecteurs - disparaissent de la population. Le rôle premier de la mutation est donc de pallier à cet inconvénient. Prise isolément, la mutation constitue une exploration aléatoire de l'espace des chaînes. Mais une utilisation parcimonieuse avec la reproduction et le croisement constitue une police d'assurance protégeant de la perte prématurée de composantes importantes.

Du rôle secondaire de l'opérateur de mutation dans l'AGC, nous retiendrons que la fréquence de mutation conseillée pour obtenir de bons résultats dans les études empiriques se situe autour d'une mutation tous les 1000 bits (positions). Les taux de mutation sont également faibles dans les populations naturelles, ce qui nous conduit à penser que la mutation est considérée à juste titre comme un mécanisme d'adaptation sécondaire pour les AG.

Nous illustrons à la figure 6, pour un codage binaire, un schéma de mutation.

Un rapprochement plus profond des AG à la biologie moléculaire a permis la mise en évidence d'autre opérateurs génétiques et d'autres mécanismes de reproduction.

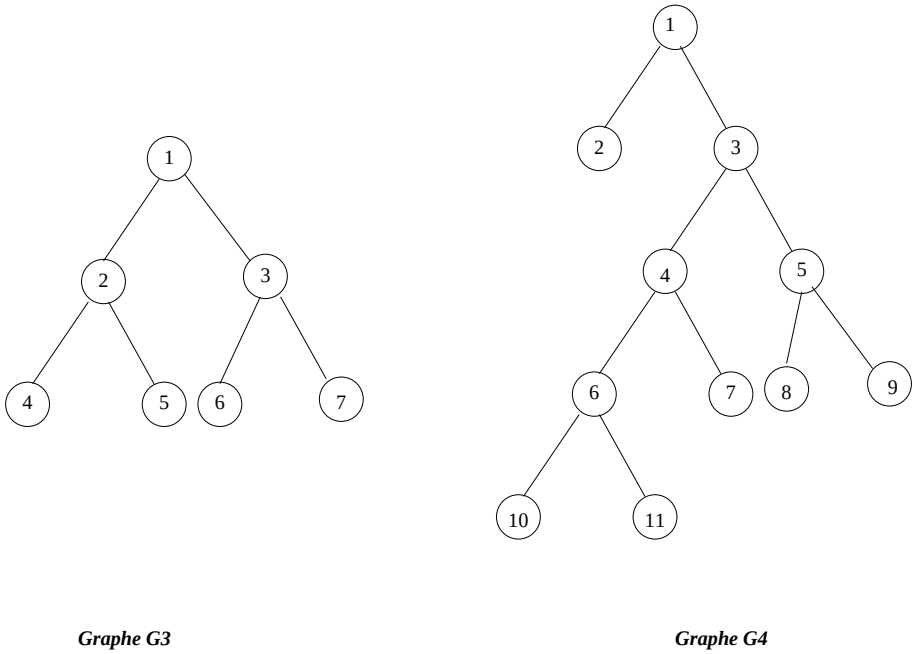


FIG. 4 - *Graphes issus du croisement*

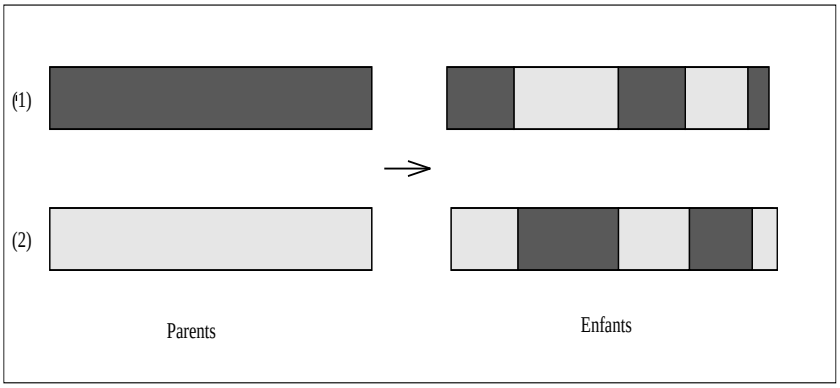


FIG. 5 - *Croisement uniforme*

On peut citer entre autre, l'inversion, la combinaison de l'inversion et du croisement, etc. Ces opérateurs ont été utilisés avec succès dans plusieurs contextes [11, 1987]. Cependant, les trois opérateurs O1, O2 et O3 que nous avons étudiés dans cette

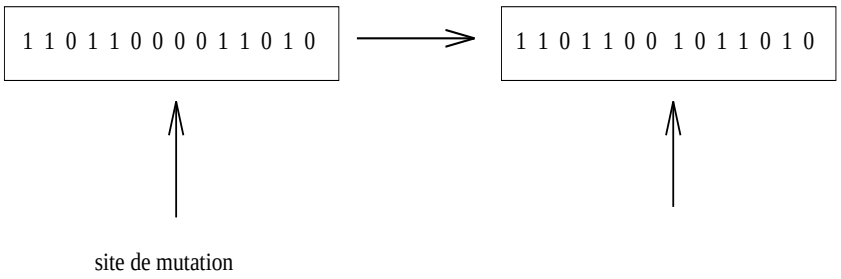


FIG. 6 - *Mutation du caractère 0 en 1*

partie ont fait la preuve de leur simplicité de mise en oeuvre algorithmique et de leur efficacité pour résoudre un nombre important de problèmes d'optimisation par les AG.

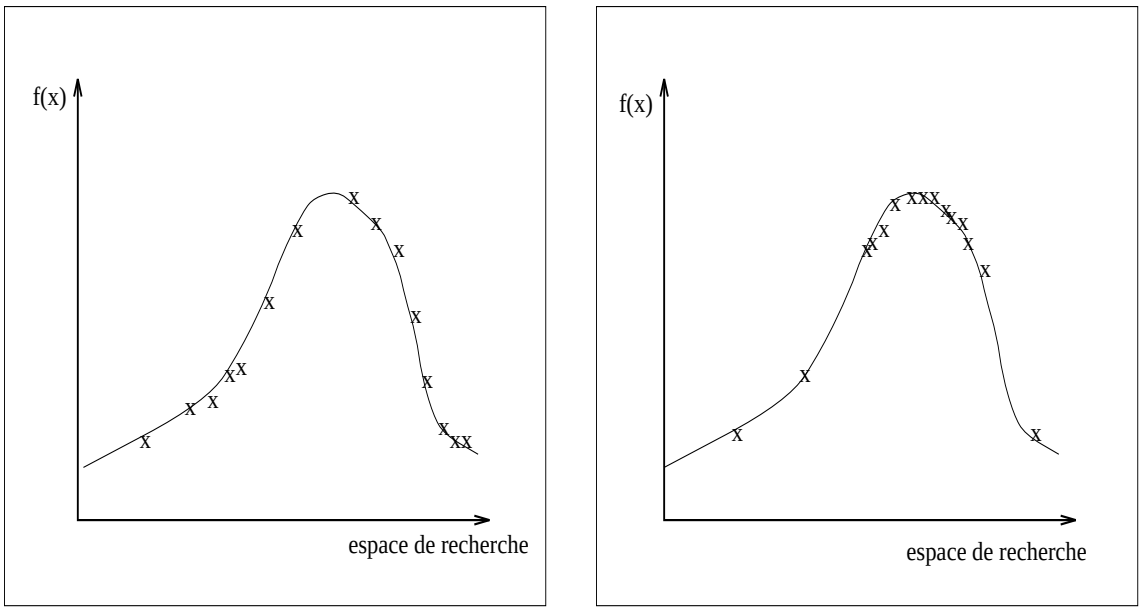


FIG. 7 - *Evolution de la population*

2.2.2 Les paramètres

Les opérateurs de l'AGC sont guidés par un certain nombre de paramètres généralement fixés à l'avance et dont dépend très fortement la "bonne" convergence de l'algorithme. Présentons brièvement les principaux paramètres de l'AGC.

- ♣ La taille de la population et la longueur du codage de chaque individu. Il est conseillé de prendre comme taille de la population la valeur correspondant à la longueur du codage des individus. En effet, si la taille de la population est très grande, l'évaluation de tous les individus de la population peut s'avérer trop longue. Par contre, si elle est très petite, l'algorithme peut converger trop rapidement.

- ♣ la probabilité de croisement p_c. Elle dépend de la forme de la fonction d'évaluation. Son choix est généralement expérimental et sa valeur est très souvent prise entre 0.5 et 0.9. Plus elle est élevée, plus la population subit des changements importants. Ainsi, la convergence est très rapide si le taux de croisement est proche de 1.

- ♣ la probabilité de mutation p_m . Le taux de mutation est généralement faible, le risque d'un taux élevé étant de modifier les meilleurs individus et ainsi, de s'éloigner de l'optimalité.

Il ressort clairement des divers détails donnés sur les paramètres de l'AGC que la fixation des différentes valeurs numériques des précédents paramètres, reste une tâche assez complexe. En fait, la délicate gestion des différents paramètres de l'AGC n'est qu'une traduction de la difficile maîtrise de ceux-ci au niveau moléculaire. A ce propos, voici quelques remarques d'un biologiste [38, 1987], au sujet du taux de la mutation.

"... Le taux de mutation est généralement très faible et pour obtenir une estimation précise, il est nécessaire d'étudier un grand nombre de gènes. Les généticiens classiques ont déterminé que le taux de mutation qui change des caractères phénotypiques ou provoque des effets mortels est de l'ordre de 10^{-5} par locus par génération dans les organismes supérieurs comme l'homme, la drosophile et le maïs. Cependant, on en sait très peu sur la nature de ces mutations au niveau de l'ADN. En outre, la plupart de ces mutations sont délétères et contribuent peu aux changements dans l'évolution des organismes. Récemment, il y a eu de nombreuses tentatives de mesurer le taux de mutation au niveau moléculaire. En étudiant 7 loci d'enzyme et 3.111.598 générations de gènes (nombre de gènes examinés $\times$ nombre de générations), Voelker et al [45, 1980] ont estimé que les taux des changements de mobilité

électrophorétique et des mutations nulles étaient de 1.28×10^{-6} et 3.86×10^{-6} par locus et par génération, respectivement."

2.2.3 Génération d'une nouvelle population

Nous avons décrit un ensemble suffisant d'opérateurs standards dont se sert le programmeur pour générer les nouveaux individus à partir des descendants de la population courante. Ces opérateurs peuvent constituer une étape directe ou indirecte dans le protocole de création d'une nouvelle génération d'individus. Ainsi , la littérature [35, 1994] des AG offre une collection fort variée de stratégies dont il convient ici d'en citer quelques unes:

(1) la nouvelle population est composée uniquement des enfants. On laisse alors disparaître tous les individus de la population courante. L'inconvénient majeur de cette approche est le risque de perdre le meilleur individu lorsqu'on ne le conserve pas automatiquement dans la nouvelle population.

(2) les enfants remplacent d'une façon régulière les individus les moins forts de la génération courante.

(3) la nouvelle génération est constituée des n meilleurs individus de la population intermédiaire formée des enfants et des parents.

2.3 Le théorème fondamental des A.G.

Les Algorithmes génétiques font intervenir constamment des opérateurs aléatoires. Cependant, l'influence de l'arbitraire du choix est bien moindre qu'on peut avoir tendance à le croire. Afin de bien comprendre le processus de fond - appelé théorème fondamental - qui guide l'algorithme génétique, introduisons la notion de schéma.

2.3.1 Les schémata

Une bonne partie des études menées sur les AG repose sur leur approche par information globale. Ainsi, plusieurs auteurs se sont intéressés aux similarités entre les individus d'une population. La formalisation de cette recherche de similarités entre les individus utilise des méta-symboles ou des sous ensembles particuliers de

l'espace de recherche appelé *schéma*[4]. Cette notion se retrouve d'ailleurs dans les travaux de I.C. Lerman sous le nom de *Cylindre à base ponctuelle* [31, 1991].

Définition 2.1 *Soit* $X = X_1X_2...X_n$ *une séquence de longueur* n *sur l'alphabet binaire* $\{0, 1, *\}$ *où* $*$ *est une indéterminée booléenne susceptible de prendre l'une des deux valeurs logiques* 0 *ou* 1. *Une chaîne binaire* $b = b_1b_2...b_n$ *est dite instance du motif* X *si et seulement si*

$$\forall i \in \{1...n\} \quad b_i = X_i \text{ ou } b_i \in \{0, 1\} \text{ si } X_i = *$$

Définition 2.2 *L'ensemble des chaînes binaires de longueur* n *qui sont des instances d'un motif spécifié* S *de longueur* n *est appelé schéma .*

Par exemple, le schéma $S = 10*10*0$ représente l'ensemble des individus suivants classés dans l'ordre lexicographique.

$$S^1 = 1001000$$

$$S^2 = 1001010$$

$$S^3 = 1011000$$

$$S^4 = 1011010$$

Relativement à la notion de schéma, introduisons deux caractéristiques couramment utilisées dans la théorie des AG.

- L'*ordre* d'un schéma S, noté $o(S)$, correspond au nombre de positions instanciées dans le schéma S.
- La *longueur de définition* , notée $l(S)$, désigne la distance entre la première et la dernière position instanciée du schéma S [cf. figure 8].

Ainsi, dans l'exemple ci-dessus $S = 10 * 10 * 0$ a pour ordre $o(S) = 5$, et pour longueur de définition $l(S) = 7$.

4. Au pluriel, schémata.

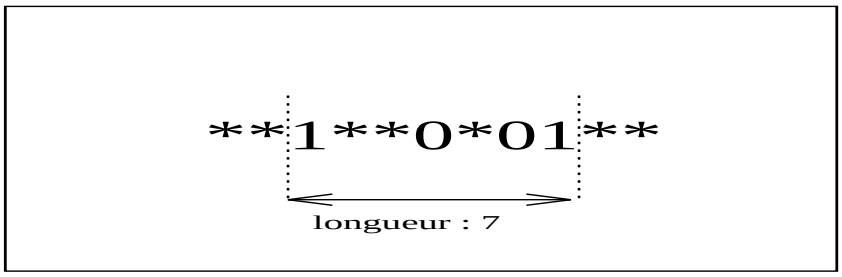


FIG. 8 - *Longueur de définition d'un schéma*

2.3.2 Théorème des schémata

Nous considérons un AG canonique opérant sur une fonction f définie sur le cube logique $\mathbb{B} = \{0,1\}^N$ - N entier naturel - à valeurs dans $\mathbb{R}_+^*$. Désignons par P une population d'individus - chaînes binaires - de longueur N. À $\mathbb{B}$, on associe la suite des valeurs

$$\{f(w) \ / \ w \in \mathbb{B} \ \}$$

Définition 2.3 *L'utilité d'un schéma S vis à vis de la population P se mesure par l'expression:*

$$u(P,S) \ = \ \frac{1}{|S \cap P|} \sum_{w \in S \cap P} f(w)$$

où $|S \cap P|$ désigne le cardinal de l'ensemble $S \cap P$.

Désignons par (P_t) $t \in \mathbb{N}$ les différentes populations successives obtenues par l'AGC. Le théorème fondamental des algorithmes génétiques indique que la fréquence relative de présence de représentants dans la population P_t, appartenant à un schéma S a d'autant plus tendance à augmenter - dans la population P_{t+1} - que l'utilité de S est meilleure. Formellement, ce théorème se traduit par l'inégalité suivante [24, 1975].

Théorème 2.1 (des schémata) *Si (P_t) $t \in \mathbb{N}$ désigne les populations successives obtenues par l'AGC, alors*

$$E([S \cap P_{t+1}]) \geq |S \cap P_t| \frac{u(P_t,S)}{u(P_t,\mathbb{B})}(1 \ - \ c(P_t,S) \ - \ m(P_t,S))$$

où E désigne l'espérance mathématique et c(.) (resp. m(.) approximent les probabilités qu'une instance de S soit détruite - n'apparaisse pas dans la génération suivante - par l'opérateur de croisement (resp. mutation).

Ce résultat peut se formuler en utilisant les différents paramètres de l'AGC et les caractéristiques du schéma S. Pour le croisement, nous nous limitons au croisement à un seul site. En effet, la probabilité qu'un schéma S de taille N soit détruit - n'est plus de représentant dans la génération - est quantifiée par

$$\frac{l(S)}{(N-1)} \tag{1}$$

où $l(S)$ est la longueur de définition du schéma S. Ainsi, un schéma ayant une grande longueur de définition a plus de chance d'être détruit par l'opérateur de croisement.

Le croisement étant appliqué avec une probabilité p_c, la survie du schéma S après croisement simple - un seul site de croisement - est alors minorée par la probabilité

$$1 - \left(p_c \frac{l(S)}{(N-1)}\right) \tag{2}$$

D'autre part, un chromosome du schéma peut subir des changements sous l'effet de la mutation. Le schéma restera inchangé si aucune des $o(S)$ positions instanciées ne subit l'influence de l'opérateur de mutation. Ainsi, la destruction du schéma S par une mutation de probabilité - petite -p_m se traduit par la probabilité

$$(1-p_m)^{o(S)} \approx 1 - o(S)p_m \tag{3}$$

En définitive nous avons

$$c(P_t, S) = p_c \frac{l(S)}{(N-1)}$$

et

$$m(P_t, S) = o(S)p_m$$

Dans ces conditions, l'inégalité du théorème des schémata se présente sous la forme

$$E([S \cap P_{t+1}]) \geq |S \cap P_t| \frac{u(P_t, S)}{u(P_t, \mathbb{B})} (1 - p_c \frac{l(S)}{(N-1)} - o(S)p_m)$$

A travers cette reformulation, on peut remarquer que les *building blocks* - schémata de courte longueur de définition et de faible ordre - auront tendance à se multiplier considérablement dans la population. Ainsi, la bonne convergence de l'AG reposerait sur la capacité de recombiner ces building blocks par croisement pour obtenir des chromosomes de plus en plus performants. L'utilité relative des schémata $u(P_t, S)$ devient alors un aspect fondamental de la mesure de performance d'un algorithme génétique [19, 1990] [3, 1991] [14, 1991] [22, 1992].

2.3.3 Parallèlisme implicite

Lorsqu'on regarde l'AG comme un méga-opérateur sur l'espace des schémata, une des conséquences pratiques du théorème des schémata est le phénomène appelé *parallélisme implicite*[5] des opérations. En fait, chaque évaluation d'un individu fournit une information sur les schémata qu'il représente. Une population de m chromosomes engendre entre 2^N et $m * 2^N$ schémata. Le nombre de schémata traités est en moyenne de l'ordre de m^3. Ainsi, plus large est l'espace de recherche, plus intéressant est l'emploi d'un AG par rapport à d'autres technique [10, 1990]. Le parallélisme implicite peut se résumer dans l'énoncé plus précis suivant [2, 1993]:

Théorème 2.2 (Sur le parallèlisme implicite) *Soit P une population de taille $m = \alpha 2^l = 2^{\beta l}$ dont les individus sont choisis indépendamment et uniformément de façon aléatoire dans $\mathbb{B} = \{0,1\}^N$. $\alpha \geq 1,\ \beta \geq 1$.*
Notons $p_\varepsilon = \frac{2l}{N}$, alors pour $1 \leq \beta \leq \frac{4}{3}$, le nombre moyen de schémata disjoints qui se propagent avec une probabilité inférieure à p_ε d'être détruit par application de l'opérateur de croisement est au moins de :

$$\left(\begin{array}{c} 2l \\ \beta l \end{array} \right) 2^{\beta l}(1 - e^{-l})$$

Cette expression nous donne une borne inférieure d'ordre m^3 pour $\beta = 1$ et une borne inférieure d'ordre $m^{2.377}$ pour $\beta = \frac{4}{3}$.

En outre, avec une probabilité supérieure à $1 - 2e^{-l}$, le nombre de schémata qui se propagent est au moins

$$\frac{1}{2} \left(\begin{array}{c} 2l \\ \beta l \end{array} \right) 2^{\beta l}(1 - e^{-l})$$

5. Terminologie emprunté à Goldberg

2.3.4 Codage des solutions

Dans un Algorithme génétique, il est nécessaire, compte tenu de la complexité des objets manipulés, de travailler à un niveau synthétique, notamment pour les deux opérations fondamentales que sont le croisement et la mutation. Les solutions sont alors représentés sous forme de codes qui tiennent compte, au mieux, de la nature du problème posé. Pour notre problème de la MDS nous codons, comme nous le verrons plus loin, une configuration directement par un vecteur de réels. Cependant, les premières générations des AG, en particulier l'AGC, utilisaient essentiellement l'alphabet binaire conformément aux travaux de Holland [24, 1975] qui montre, avec la théorie des schémata, que le code optimal pour un AG est un code ayant un alphabet le plus petit possible. Dans ce contexte, le décodage d'une chaîne binaire $s = (s_0, ..., s_n)$ en un nombre réel $x \in [a, b]$ est directe et complètement déterminée par les deux opérations suivantes

(1) La séquence binaire s est convertie en base 10;

$$\begin{aligned}(s_0, ..., s_n)_2 &= (\sum_{i=0}^{n} s_i 10^i)_{10} \\ &= x'\end{aligned}$$

(2) le nombre réel x correspondant est:

$$x = a + x' \frac{b-a}{2^{n+1}-1}$$

Ces opérations sont effectuées un grand nombre de fois - à chaque évaluation d'un individu - au cours de l'évolution de l'algorithme. Par conséquent, le choix d'un code doit tenir compte de la complexité algorithmique du processus de codage/décodage dont la lourdeur peut ralentir les calculs et influencer considérablement la convergence de l'algorithme tout entier.

2.4 Convergence de l'AGC

La convergence vers une solution optimale d'un algorithme est un but toujours recherché et rarement atteint. La question se pose de façon fondamentale pour les Algorithmes génétiques et notamment l'AGC. Dans cette partie, les propriétés des chaînes de Markov sont exploitées pour étudier la convergence de l'Algorithme génétique canonique. Nous adoptons l'approche récente de Günter [21, 1994] dans

laquelle la séquence des populations produites par l'AGC est considérée comme une chaîne de Markov homogène.La propriété fondamentale d'un système markovien est *l'absence de mémoire* : l'évolution future du système ne dépend que de l'instant présent et non de son évolution passée. Toutefois il existe d'autres approches plus ou moins convaincantes. En particulier, le parallélisme implicite et le théorème des schémata ont servi d'outils de l'analyse de la convergence des AG. Malheureusement, les arguments combinatoires très immédiats utilisés dans cette analyse ont paru peu rigoureux [12, 1991]. La thèse de Cerf [5, 1994] étudie également une théorie asymptotique des algorithmes génétiques fondée sur la théorie de Freidlin-Wentzell des petites perturbations aléatoires de systèmes dynamiques.

2.4.1 Chaînes de Markov finies

Dans cette partie, nous rappelons les résultats principaux relatifs aux processus markoviens dont nous chercherons à montrer l'intérêt pour la convergence des Algorithmes génétiques dans le paragraphe suivant.

Nous considérons une chaîne de Markov homogène d'espace fini d'états E de cardinal r complètement définie par la donnée d'une part, de ses probabilités initiales $q_i = P(X_0 = E_i),\ 1 \leq i \leq r$ et d'autre part, de sa transition *matrice de transition* $P = (p_{ij})\ 1 \leq i, j \leq r$.

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{pmatrix}$$

Définition 2.4 *Un état i est dit apériodique si*

$$pgdc\{k \ / \ p_{ii}^k > 0\} \ = \ 1$$

où p_{ii}^k - coefficient de P multiplié par lui même k fois - est la probabilité conditionnelle de revenir à l'état i en k étapes.

Définition 2.5 *Une chaîne de Markov irréductible - tout état peut être atteint à partir de tout autre état - et ayant tous ses états apériodiques est dite ergodique.*

Nous donnons ci-après une liste de propriétés matricielles fondamentales partagées par les matrices de transition. La matrice $M = (a_{ij})$ référencée dans les énoncés est supposée carrée et d'ordre n.

Définition 2.6 (Propriétés des matrices de transition)

(1) *M est dite positive ($M > 0$) si* $a_{ij} > 0 \quad \forall i, j \in \{1, ..., n\}$;

(2) *M est dite non négative ($M \geq 0$) si* $a_{ij} \geq 0 \quad \forall i, j \in \{1, ..., n\}$;

Une matrice M non négative est dite :

(3) *primitive, s'il existe un entier naturel k tel que M^k soit positive. En particulier, toute matrice positive est une matrice primitive;*

(4) *réductible si elle peut être mis sous la forme (U et V sont des matrices carrées) :*

$$\begin{pmatrix} U & 0 \\ R & V \end{pmatrix}$$

en appliquant les mêmes permutations sur les lignes et les colonnes de la matrice M. Une matrice réductible n'est donc pas primitive.

(5) *stochastique si*

$$\forall i, \quad \sum_{k=1}^{r} a_{ik} = 1$$

En particulier toutes les matrices de transitions sont stochastiques et tout produit de matrices stochastiques donne une matrice stochastique. Une matrice M stochastique est dite :

(6) *stable si toutes ses lignes sont identiques;*

(7) *positive par colonne si elle a au moins un élément positif par colonne.*

Lemme 2.1 *Soient $A = (a_{ij}), B = (b_{ij})$ et $C = (c_{ij})$ $i, j \in \{1, ..., n\}$ trois matrices stochastiques avec B positive et C positive par colonne. Alors, ABC est une matrice positive.*

Théorème 2.3 *[26, 1980] Pour toute matrice M stochastique primitive, M^k - produit matriciel - converge, lorsque $k \to \infty$, vers une matrice stochastique positive et stable $M^\infty = \mathbb{1}' q^\infty$ où*

$$\mathbb{1} = (1, ..., 1)$$

et

$$q^\infty = q^0 lim_{k\to\infty} M^k = q^0 M^\infty$$

est sans élément nul, unique, et indépendante de la distribution initiale q^0.

Théorème 2.4 *[26, 1980] Soit une matrice*

$$M = \begin{pmatrix} U & 0 \\ R & V \end{pmatrix}$$

stochastique et réductible, où la matrice carrée U d'ordre l est stochastique primitive et $R, V \neq 0$. Alors :

$$M^\infty = lim_{k\to\infty} M^k = lim_{k\to\infty} \begin{pmatrix} U^k & 0 \\ \sum_{i=0}^{k-1} V^i R U^{k-i} & V^k \end{pmatrix} = \begin{pmatrix} U^\infty & 0 \\ R^\infty & 0 \end{pmatrix}$$

est une matrice stochastique stable avec $M^\infty = \mathbb{1}' q^\infty$, unique et indépendante de la distribution initiale. q^∞ satisfait : $q_i^\infty > 0$, $\forall\ 1 \leq i \leq l$ et $q^\infty = 0$, $\forall\ l < i \leq n$

2.4.2 Analyse de l'AGC par les Chaînes de Markov

Le comportement de l'Algorithme génétique canonique optimisant une fonction f définie sur le cube logique $\mathbb{B} = \{0,1\}^N$ - N entier naturel - à valeurs dans $\mathbb{R}_+^*$ est modélisé, dans ce paragraphe, par une chaîne de Markov homogène finie pour lequel l'espace d'états est donné par $\mathbb{E} = \mathbb{B}^{N.l}$, où N désigne la taille d'un chromosome et l l'effectif de la population [16, 1987]. Chaque état est alors constitué rigoureusement d'une population et d'une seule. Dans ces conditions, nous pouvons désigner par π_k^i la projection qui retourne pour l'état i le k ième chromosome de taille l . Cette fonctionalité nous permet d'identifier tous les individus d'une population-état.

Nous considérons une décomposition de la matrice P de transition de l'AGC en un produit naturel de matrices stochastiques $P = C * M * S$ où C ,M et S décrivent les transitions intermédiaires issues respectivement du croisement, de la mutation et

de la reproduction-sélection. Cette décomposition est toujours possible tel que nous l'indique la propriété 2.6. Ceci nous conduit au résultat suivant :

Théorème 2.5 *[21, 1994] Soit un AGC avec une probabilité de croisement* $p_c \in [0,1]$*, une probabilité de mutation* $p_m \in (0,1)$ *, une sélection proportionnelle et une matrice de transition* P*. Alors* P *est primitive.*

Preuve L'opérateur de croisement peut être vu comme une fonction aléatoire définie sur $\mathbb{E}$ à valeurs dans $\mathbb{E}$. Chaque état est donc lié d'une façon probabiliste à un autre état. Ainsi, C est une matrice stochastique. Ce raisonnement tient pour M et S. La probabilité qu'un état i devienne l'état j après une mutation est donnée par

$$m_{ij} = p_m^{H_{ij}}(1-p_m)^{N.l-H_{ij}} \ > 0 \quad \forall i,j \in \mathbb{E}$$

où H_{ij} désigne la distance de Hamming entre les représentations binaires des états i et j. M est donc positive.

La probabilité p_s^{ii} qu'une sélection ne perturbe pas un état généré par la mutation est bornée par

$$p_s^{ii} \geq \frac{f(\pi_1^i) \times ... \times f(\pi_{N.l}^i)}{(f(\pi_1^i) + ... + f(\pi_{N.l}^i))^{N.l}} > 0$$

pour tout état $i \in \mathbb{E}$. Ainsi S est positive par colonne. Le théorème 2.1 nous indique que dans ces conditions $P = C * M * S$ est positive. Or toute matrice positive est primitive, donc P est primitive.

Corollaire 2.1 *L'AGC avec les paramètres du théorème précédent est une chaîne de Markov ergodique.*

Preuve Conséquence immédiate des théorèmes 2.3 et 2.4.

La distribution initiale n'a donc pas d'effet sur le comportement limite de la chaîne de Markov. Ainsi l'initialisation de l'algorithme génétique canonique peut être arbitraire et l'opération de reproduction-sélection peut - théoriquement - être omise, puisque la distribution limite restera inchangée. Par contre, la propriété d'ergocité a des conséquences considérables sur le comportement asymptotique de l'AGC. Précisons dans quel sens nous entendons la convergence de l'AG.

L'objectif d'un AGC est de localiser l'ensemble

$$f^* = max\{f(x)/x \in \{0,1\}^N\}$$

des maxima globaux de f. En gardant les notations précédentes, désignons par

$$X_t = max\{f({\pi_k}^t(i))\ tq\ k = 1,...N\}$$

une suite de variables aléatoires représentant les meilleurs fitness dans la population i au temps t. Alors,

l'AGC converge vers l'optimum global si et seulement si

$$lim_{t\to\infty} P\{X_t = f^*\} \ = \ 1.$$

Ceci nous conduit à un résultat qui fait encore couler beaucoup d'encre

Théorème 2.6 *Tout AGC avec une probabilité de croisement $p_c \in [0,1]$, une probabilité de mutation $p_m \in (0,1)$, et une sélection proportionnelle ne converge pas vers l'optimum global.*

Preuve Soit un état $i \in \mathbb{E}$ tel que $max\{f(x)/x \in \{0,1\}^N\} < f^*\}$ et la probabilité ${p_i}^t$ que l'AG soit à l'état i au temps t . On a clairement,

$$P\{X_t \neq f^*\} \geq {p_i}^t \quad \leftrightarrow \quad P\{X_t = f^*\} \leq 1 - {p_i}^t$$

D'après le théorème 2.3, la probabilité que l'AG soit à l'état i converge vers ${p_i}^\infty > 0$. Par conséquent :

$$lim_{t\to\infty} P\{X_t = f^*\} \ \leq \ 1 - {p_i}^\infty \ < 1.$$

Dans la pratique, nous ne nous conformerons pas strictement au modèle aléatoire de la sélection proportionnelle; puisque la meilleure solution obtenue à une étape de l'AGC est gardée pour l'étape suivante. Ainsi, la solution globale est forcement visitée après un nombre fini, certes très élevé, de transitions. Cette technique est appelé couramment *élitisme* [17, 1989].

3 Adaptation des AG à la représentation euclidienne

3.1 Préambule

Dans cette partie, nous exploitons la souplesse des AG, décrits dans le paragraphe précédent, pour développer de nouveaux algorithmes destinés à la représentation euclidienne d'une matrice de proximités. Ce problème est mathématiquement

représentée - indépendamment des approches (solution classique, moindres carrés ou maximum de vraisemblance) - par l'équation fondamentale de l'optimisation numérique sans contrainte en dimension finie [28, 1964] [42, 1981] [29, 1984]

$$f_0 \;=\; min\{f(x) \;/\; x \in \mathbb{R}^k\}$$

où f désigne la fonction coût adoptée et k la dimension de la configuration recherchée.

Bien que les critères, [cf. table 1], dépendent des méthodes, nous nous efforçons à proposer des procédures valides, pour toutes les fonctions coût, basées sur les Algorithmes génétiques.

Méthode	*Fonction coût*
Torgerson-Gower (PCA ou AFTD)	$tr(W - XX')^2$
Kruskal-Shepard	$\sum_{i<j}(\hat{\delta_{ij}} - d_{ij}(X))^2$
Leeuw-Heiser	$\sum_{i<j}(\delta_{ij} - d_{ij}(X))^2$
Takane-Young	$\sum_{i<j}(\delta_{ij}^2 - d_{ij}^2(X))^2$
Ramsay	$\sum_{i<j}(log\delta_{ij} - log d_{ij}(X))^2$

TAB. 1 - *Quelques critères utilisés en MDS*

La table 1 est construite à partir des articles [13, 1986],[37, 1992] et [9, 1994] et donne un aperçu des fonctions coût existantes dans la littérature de la MDS. Elle utilise les notations suivantes:

δ_{ij} dissimilarité entre les objets i et j,

X configuration dans un espace de dimension k,

d_{ij} distance entre les objets i et j dans X,

W matrice dite de Torgerson,

$\hat{d}$ transformation monotone croissante optimale de δ :*disparité*[6],

6. Traduction du mot americain *disparities*

PCA Principal Coordinate Analysis.

Nous considérons deux directions dans la présentation des résultats. La première est conforme à la vision classique de la MDS (PCA ou AFTD) où la solution se trouve défini par des facteurs orthogonaux $[u^1, ..., u^k]$ dont chacun peut être interpreté comme une fonction sur l'ensemble des sommets du nuage auquel il est relatif. La deuxième direction présente directement la configuration issue de l'optimisation des fonctions coût. Ainsi l'individu de AG est directement lié à la nature mathématique de l'objet recherché: les configurations spatiales de points.

Sans distinguer rigoureusement ces deux directions, nous organisons ce paragraphe comme suit. Nous commençons par décrire au paragraphe 3.2 le fitness et l'espace de recherche de nos algoritmes. C'est au paragraphe 3.3 que nous présentons notre codage qui a un caractère très synthétique et géométrique. Les opérateurs de croisement et de mutation adaptés à nos individus sont ensuite présentés au paragraphe 3.4. Le paragraphe 3.5 est réservé à la description des algorithmes que nous proposons. Nous montrerons au paragraphe 3.6 comment accélérer nos AG par hybridation en combinant à ceux-ci les algorithmes classiques d'optimisation. Nous terminons par la présentation de la fonction dite de partage - *sharing* - qui permet d'avoir un ensemble de solutions dans le cas des fitness multimodaux.

Nous n'aborderons ni les méthodes de collecte de l'information conduisant à la construction des matrices de proximités , ni l'étude du caractère euclidien de ces matrices [39, 1993].

3.2 Fitness et espace de recherche

Nous supposons la donnée d'une fonction coût positive, à minimiser, f quelconque [cf. Table 1].

Si nous notons par X un individu de l'AG, le fitness, quantité positive (cf. §2), est donnée, par

$$\tilde{f}_i = -f(X_i) + max_{1 \leq i \leq n}\{f(X_i)\} + \varepsilon$$

où ε est une constante positive très petite. L'ajout de cette constante permet d'attribuer une probabilité de sélection non nulle à l'individu le plus faible.

L'espace de recherche, d'un AG classique, dans un problème d'optimisation d'une fonction de n variables est une boule hypercubique $[a_1, b_1] \times ... \times [a_n, b_n]$ de $\mathbb{R}^n$. Ce qui suppose - pour une bonne efficacité numérique - une identification a priori de la zone solution. Par ailleurs, notre problème possède un ensemble de solutions pouvant décrire un cône. En effet, si Y est une configuration optimale, la configuration αY

est aussi une configuration optimale. En d'autres termes, la fonction coût de la MDS est invariante par toute transformation homothétique . Cette invariance s'étend d'ailleurs à toutes les transformations appartenant au groupe des similitudes[7].

Dans ces conditions, une boule de l'espace de représentation - de dimension deux ou trois - munie d'une subdivision discrète est suffisante comme espace de recherche. Ainsi, nous considérons pour espace de recherche, dans le cas d'une représentation planaire, un carré de côté égal à l'unité.

3.3 Représentation chromosomique: codage des individus

Classiquement pour l'emploi d'un AG en optimisation, le chromosome est directement la concaténation des codes binaires des composantes du vecteur représentant un point de l'espace de recherche. Dans notre problème, une solution de la MDS ou de l'AFTD dans un espace de dimension $k < n$ est une matrice $n \times k$ X de réels, de la forme suivante:

$$X = \begin{bmatrix} x_{11} & x_{1k} \\ \vdots & \\ x_{n1} & x_{nk} \end{bmatrix}$$

Ainsi, une population de solutions dans l'espace de recherche est un ensemble de matrices réelles $M(n \times k)$. Nous adoptons un code directement lié a la matrice et donné, pour l'individu i, par le vecteur réel de taille $n * k$

$$X_i = \left(x^i_{11}, ..., x^i_{1k}, x^i_{21}, ..., x^i_{2k}, ..., x^i_{n1}, ..., x^i_{nk}\right)$$

où les x^i_{lj} sont des réels et le sous-ensemble $x^i_{j1}, ..., x^i_{jk}$ correspond aux coordonnées de la j-ème entité traitée.

L'intérêt majeur, par rapport au codage binaire, de ce type de codage réside dans sa capacité réductrice de la complexité. En effet, le décodage - porté directement sur les indices - est quasi-immédiat et ne nécessite aucunement une opération de codage/décodage semblable à celle que nous avons présentée au paragraphe 2.3.4.

Nous restreignons, dans la suite de l'article, nos analyses sur un espace de dimension 2, ce qui n'altère en rien la perspicacité de nos résultats.

3.4 Croisement et Mutation

Etant donné deux configurations codées $X_1 = (x^1_1, ..., x^1_{nk})$ et $X_2 = (x^2_1, ..., x^2_{nk})$ un croisement classique, que nous appelons par la suite *croisement global* produit des

7. Rotation, symétrie, translation, homothétie

nouvelles configurations suivant le principe énoncé au paragraphe 2.2.1. Le croisement uniforme correspond, ici, à interchanger un sous ensemble aléatoire de sommets [cf. figure 9], alors q'une opération de mutation concernant une même configuration correspond à un déplacement " adéquat " d'un sommet choisi aléatoirement. Par exemple, en considerant les individus parents de la figure 11:

$$
\begin{aligned}
X &= (x1, x2, x3, x4, x5, x6, x7) && (4)\\
&= (x1_1, x1_2, x2_1, x2_2, x3_1, x3_2, x4_1, x4_2, x5_1, x5_2, x6_1, x6_2, x7_1, x7_8) && (5)\\
Y &= (xa, xb, xc, xd, xe, xf, xg) && (6)\\
&= (xa_1, xa_2, xb_1, xb_2, xc_1, xc_2, xd_1, xd_2, xe_1, xe_2, xf_1, xf_2, xg_1, xg_8) && (7)
\end{aligned}
$$

Un croisement uniforme qui porte sur les sites - indices - 3, 4, 7, 8, 11 et 12 produit deux individus enfants:

$$
\begin{aligned}
X' &= (x1, xb, x3, xd, x5, xf, x7) && (8)\\
&= (x1_1, x1_2, xb_1, xb_2, x3_1, x3_2, xd_1, xd_2, x5_1, x5_2, xf_1, xf_2, x7_1, x7_8) && (9)\\
Y' &= (xa, x2, xc, x4, xe, x6, xg) && (10)\\
&= (xa_1, xa_2, x2_1, x2_2, xc_1, xc_2, x4_1, x4_2, xe_1, xe_2, x6_1, x6_2, xg_1, xg_8) && (11)
\end{aligned}
$$

Une difficulté émerge lorsqu'on applique, conformément à la théorie des AG, les opérateurs que nous venons de décrire avec une probabilité faible pour la mutation. Le capital génétique - l'ensemble des composantes de tous les vecteurs - des différentes générations reste pratiquement inchangé et diffère très peu de celui de la population initiale. Ceci s'explique par le fait que le croisement a finalement pour simple rôle de changer les indices des composantes. Une première idée a été d'enchérir la probabilité de mutation pour maintenir un diversité suffisant et ainsi préserver l'amélioration. Mais les travaux de De Jong démontre clairement que cette méthode ne constitue pas la panacée. Une augmentation incontrôlée du taux de mutation peut dégrader la performance de l'algorithme génétique. Nous avons donc introduit un nouveau opérateur dit de *croisement local* qui agit localement sur des couples de composantes des deux individus à croiser. Ce croisement génératrice de nouvelles composantes se justifie par le résultat suivant:

Théorème 3.1 (Propriété sur le codage binaire) *Soient C_1 et C_2 deux individus binaires d'un AG. Si C'_1 et C'_2 sont deux individus obtenus par un croisement multi-site de C_1 et C_2, l'égalité suivante est vérifiée*

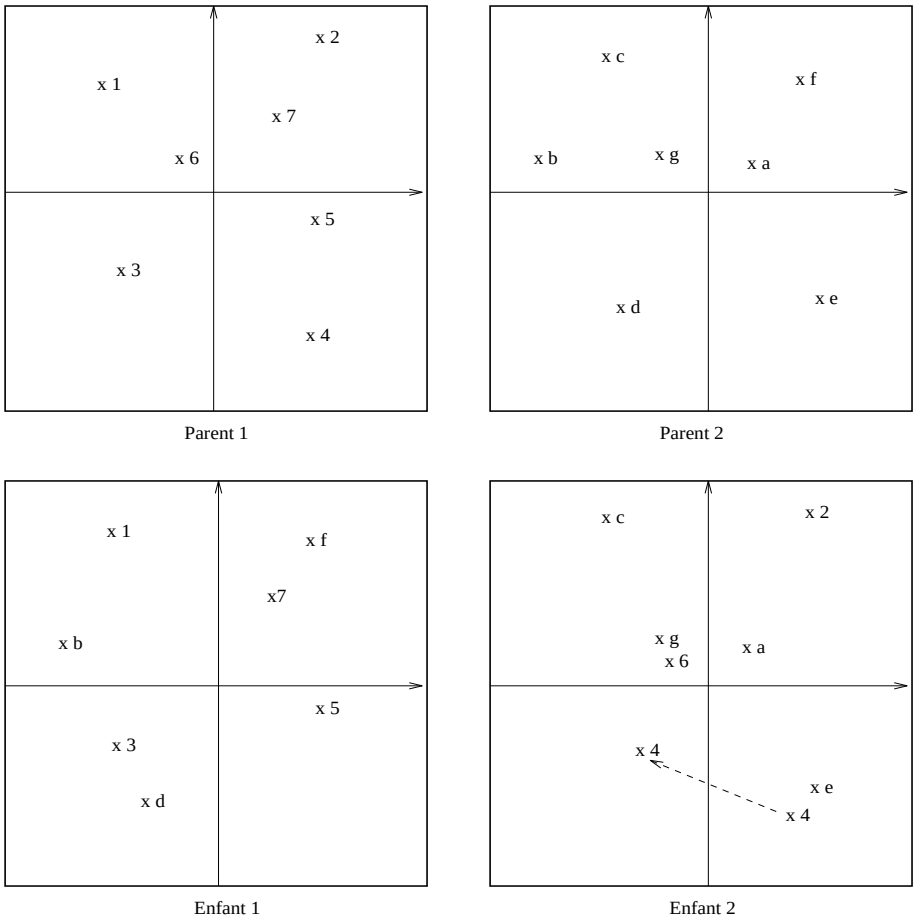


FIG. 9 - *Croisement de deux configurations planaires*

$$C'_1 + C'_2 \;=\; C_1 + C_2$$

En d'autres termes, la somme des enfants issus d'un croisement multi-site est toujours égale à la somme des parents.

Par exemple, si nous considérons deux chromosomes C_1 et C_2

$$C_1 \;=\; 1011100010101$$
$$C_2 \;=\; 0010100100111$$

Un croisement multi-site (3,7,10) donne

$$C'_1 = 0011100100101$$
$$C'_2 = 1010100010111$$

et nous avons bien

$$C_1 + C_2 = C'_1 + C'_2 = 1110000111100$$

Relativement à deux individus X_1 et X_2 le croisement local s'effectue comme suit:

on choisit aléatoirement, dans l'esprit des AG, environ 60% des couples - (x_i^1, x_i^2) - de composantes d'un couple d'individus parents tiré au hasard parmi les plus mal classés dans l'ordre de performance - typiquement 50% -. L'une des composantes, x_i^1 par exemple, du couple ainsi sélectionné est remplacée par un réel x'_1 tiré au hasard dans [a,b] où

$$a = max(x_i^1, x_i^2) - min(x_i^1, x_i^2)$$
$$b = min(x_i^1 + x_i^2, Sup)$$

et la composante x_i^2 est substituée, conformément à 3.1, par $x_i^1 + x_i^2 - x'_1$.

$Sup, Max(.)$ et $min(.)$ désignent respectivement la borne supérieure de l'espace réel de recherche, les fonctions maximum et le minimum usuelles.

En définitive notre *croisement* dit *local* est pour notre problème, une adaptation très intéressante du croisement multi-site classique sur des chromosomes produits par concaténation des codes binaires.

3.5 Description des algorithmes

L'algorithme génétique élitiste[8] ci-dessous calcule une matrice solution X du problème de la représentation affine d'une matrice de dissimilarité. Etant donné g une fonction coût et les differents paramètres, le schéma de base de cet algorithme destiné à optimiser directement la fonction coût, se déroule de façon récursive comme suit :

f fitness ou fonction d'évaluation,

8. le meilleur individu est automatiquement sélectionné pour la génération suivante

n taille de la population,

c_μ fonction de croisement. Cette fonction peut être une composée de differents types de croisement.

m_ν fonction de mutation.

Début
  $t \leftarrow 0$
  μ probabilité de croisement
  ν probabilité de mutation
  $X(t)$ population initiale
  Evaluer $X(t)$
  Répéter
    Chercher le meilleur individu $X_m(t)$
    $f(X_m(t)) = min_{1 \leq i \leq n} f(X_i(t))$
    Si $X_m(t)$ est "bon" **STOP**
      $t \leftarrow t + 1$
    Générer $Y(t)$ par *reproduction-sélection*
    $Y'(t) = c_\mu[Y(t)]$ *croisement*
    $Y''(t) = m_\nu[Y'(t)]$ *mutation*
      Si $X_m(t-1) \notin Y''(t)$
        $X(t) = (X_m(t-1), Y''(t))$
      Sinon
        $X(t) = Y''(t)$
      fsi
  jusqu'à " *bonne solution* "
Fin

Les contraintes qui s'ajoutent à ce schéma de base pour la deuxième direction - qui suit la solution classique de la MDS (PCA ou AFTD) - sont d'une part le choix de la fonction coût [cf. table 1] et d'autre part, la construction des facteurs orthogonaux solutions par la décomposition en valeurs singulières que l'on trouve

dans bon nombre de librairies scientifiques. En particulier nous pouvons citer la librairie LAPACK[9]

Définition 3.1 (Décomposition en valeurs singulières) *Soient X une matrice de taille $n \times p$ de rang r, $\{a_1, ..., a_r\}$ les vecteurs propres orthonormés de la matrice XX^t associés aux r valeurs propres non nulles rangées par ordre décroissant $\lambda_1, ..., \lambda_r$, $\{b_1, ..., b_r\}$ les vecteurs propres orthonormés de la matrice X^tX associés aux mêmes valeurs propres. La décomposition en valeurs singulières (SVD[10]) de X est donnée par:*

$$X = A\Lambda^{\frac{1}{2}}B^t$$

où

la matrice $A(n \times r)$ (resp. $B(p \times r)$) sont constituées par les vecteurs a_k (resp. b_k).

Λ est une matrice diagonale contenant les λ_k rangées par ordre décroissant.

Dans la version AFTD, les facteurs sont donnés par la SVD de la matrice X solution de l'optimisation du fitness lié à la fonction coût

$$f(X) = tr(W - XX')^2$$

Nos algorithmes calquées sur le modèle canonique de Holland - AGC - conservent, malgré notre adaptation, l'inconvénient principal des GAs qui est le temps de convergence. Dans ce qui suit nous proposons une phase d'accélération par l'introduction d'une part de nouvelles notions de croisement et d'autre part d'un nouvel opérateur que nous appelons *descente* . Cet opérateur s'inspire directement des méthodes de descente d'optimisation usuelles.

3.6 Hybridation de nos AG

Un Algorithme génétique hybride désigne un AG utilisant des algorithmes de recherche locale comme opérateurs. Ainsi, à partir d'un individu X et un voisinage $v(X)$ de X, ces opérateurs fournissent toujours un individu de qualité au moins égale à celle de X. Cette logique qui ne s'inscrit pas dans l'objectif des opérateurs de croisement et de mutation classiques se traduit par une accélération au niveau

9. La version unix est écrite en fortran 77. Sous IBM, consulter la librairie NAG écrite également en fortran 77.

10. Singular Value Decomposition

de la convergence des AG. Après avoir décrit le nouvel opérateur de croisement portant sur plusieurs individus (> 2), nous présenterons deux méthodes de descente n'utilisant pas de dérivés - Powell, Hooke et Jeeves - et pouvant être combinées aux AG. Signalons que nous utiliserons également dans nos applications les méthodes basées sur le gradient.

3.6.1 Croisement v-aire

Le type de croisement que nous proposons, dans cette partie, reprend les idées de Spendley, Hext et Himsworth [40, RAO] dans la méthode d'optimisation sans dérivés appelée *simplex method* (à ne pas confondre avec la méthode de simplexe qu'on trouve en programmation linéaire). Le simplexe étant une figure géométrique ayant $n+1$ sommets dans un espace de dimension n.

Contrairement à l'opérateur de croisement classique qui est binaire préfixé - deux opérandes - , nous avons un opérateur v-aire préfixé - v opérandes - qui se déroule comme suit:

1. On part d'un ensemble de n points (individus) $X_1, X_2, ..., X_n$; et de trois coefficients fixes: $v, \zeta et \eta$.

2. Déterminer X_h, X_l, X_0 *et* X_r

$$X_h \quad tq \quad f(X_h) = max_{1 \leq i \leq n}[f(X_i)]$$

$$X_l \quad tq \quad f(X_l) = min_{1 \leq i \leq n}[f(X_i)]$$

$$X_0 = \frac{1}{n-1} \sum_{i=1, i \neq h}^{n} X_i$$

$$X_r = (1+v)X_0 - vX_h$$

3. Remplacer X_h par X_z

   - Si $f(X_r) < f(X_l)$ construire un nouveau point, sinon passer à l'étape suivante

     $$X_e = \zeta X_r + (1-\zeta)X_0$$

     Si $f(X_e) < f(X_l)$ alors $X_z = X_e$, sinon $X_z = X_r$.

– Si $f(X_r) > f(X_i) \ \ \forall i \neq h$

– Si $f(X_r) > f(X_h)$ construire un nouveau point

$$X_c = \eta X_h + (1-\eta)X_0$$

Si $f(X_c) > f(X_h)$ remplacer tous les X_i par $\frac{(X_i+X_l)}{2}$, sinon $X_z = X_c$.

– Sinon $X_z = X_r$.

3.6.2 Méthode de Powell

Cette méthode, sans dérivées, qu'on peut rattacher à la famille des méthodes de directions conjuguées, repose essentiellement sur l'idée suivante : lorsque l'on recherche le minimum d'une fonction quadratique g successivement suivant p directions conjuguées $d_1, d_2, ..., d_p$, p plus petit que le nombre de variables de la fonction à optimiser, en partant d'un point X_0, on génère une suite $X_1, X_2, ..., X_p$ définie par :

$$g(X_i) = g(X_{i-1} + \tau_i d_i) = Min_\tau\{g(X_{i-1} + \tau d_i)\}$$

La procédure de Powell se décrit, pour toute fonction g, comme suit :

1. Choisir un point initial X_0 et p directions linéairement indépendantes $d_1, d_2, ..., d_p$. Construire une suite $X_1, X_2, ..., X_p$ de points tels que

$$g(X_i) = g(X_{i-1} + \tau_i d_i) = Min_\tau\{g(X_{i-1} + \tau d_i)\}$$

2. Poser

$$\tilde{g} = g(2X_p - X_0)$$

$$\kappa = Max_{i=1,...,n}\{g(X_{i-1}) - g(X_i)\}$$

Soit m l'indice de la direction par laquelle ce maximum est atteint.

Si $\tilde{g} \geq g(X_0)$ et/ou si

$$(g(X_0) - 2g(X_1) + \tilde{g})(g(X_0) - g(X_1) - \kappa)^2 \geq \frac{1}{2}\kappa(g(X_0) - \tilde{g})^2$$

le point de départ à l'itération suivante est X_p et les anciennes directions $d_1, d_2, ..., d_p$ sont gardées.

Si $\tilde{g} < g(X_0)$ et

$$(g(X_0) - 2g(X_1) + \tilde{g})(g(X_0) - g(X_1) - \kappa)^2 < \frac{1}{2}\kappa(g(X_0) - \tilde{g})^2$$

on recherche le minimum $\tilde{X}$ qui sera pris comme point de départ à la prochaine itération, de g dans la direction $d = X_p - X_0$. L'ordre des nouvelles directions est

$$d_1, d_2, ..., d_{m-1}, d_{m+1}, ..., d_p, d$$

Les directions initiales $d_1, d_2, ..., d_p$ ne sont pas forcément toutes conjuguées deux à deux et peuvent être définies par les axes des coordonnées.

3.6.3 Méthode de Hooke et Jeeves

Cette méthode a l'avantage d'explorer le voisinage de chaque particule visitée. La procédure générale se déroule de façon récursive comme suit :

1. Partir d'un point initial arbitraire $X_1 = (x_1, x_2, ..., x_n)$ en se donnant des pas Δx_i suivant les directions $a_1, a_2, ..., a_n$ correspondant aux axes des coordonnées. Poser k=1;

2. Calculer $g_k = g(X_k)$, poser $i = 1$ et $Z_{k0} = X_k$.

3. On perturbe les variables x_i du point intermédiaire pour générer un nouveau

$$Z_{k,i} = \begin{cases} Z_{k,i-1} + \Delta x_i a_i & \text{si} \quad g^+ < g(Z_{k,i-1}) \\ & \quad g^+ = g(Z_{k,i-1} + \Delta x_i a_i) \\ Z_{k,i-1} - \Delta x_i a_i & \text{si} \quad g^- < g(Z_{k,i-1}) < g^+ \\ & \quad g^- = g(Z_{k,i-1} - \Delta x_i a_i) \\ Z_{k,i-1} & \text{si} \quad g(Z_{k,i-1}) < min\{g^-, g^+\} \end{cases}$$

4. Si on obtient $Z_{k,n} = X_k$, réduire le pas Δx_i en divisant, par exemple par deux et revenir à l'étape précédente.

Sinon la nouvelle particule de recherche est

$$X_{k+1} = Z_{k+1}$$

5. A partir de X_k $et X_{k+1}$, établir la nouvelle direction de recherche S par

$$S = X_{k+1} - X_k$$

On construit $Z_{k+1,0} = X_{k+1} + vS$ où v est le pas qu'on peut prendre égale à 1. Une minimisation unidimensionelle peut aussi permettre de dégager une valeur optimale de v^* de v.

6. Poser $k = k + 1$, $g_k = g(Z_{k0})$, $i = 1$ et répéter l'étape 3.

Si à la fin de l'étape 3, $g(Z_{k,n}) < g(X_k)$, prendre $X_{k+1} = Y_{k,n}$ et passer à l'étape 5 .

Sinon , $g(Z_{k,n}) \geq g(X_k)$. Prendre $X_{k+1} = X_k$, réduire les pas Δx_i, poser $k = k + 1$, puis aller à l'étape 2.

7. La procédure se termine lorsque

$$max_i(\Delta x_i) < \varepsilon$$

3.7 La fonction de partage(sharing function)

Les critères d'évaluation utilisés en représentation géométrique possèdent généralement plusieurs extréma globaux. Il est très important de pouvoir détecter tous ces extréma. Or les algorithmes génétiques que nous avons présentés convergent, théoriquement, vers un seul pic. Nous présentons la technique de partage, inspirée du phénomène naturel de création des niches écologiques, qui permet une répartition pondérée des chromosomes sur tous les sommets de la fonction à optimiser. Cette technique est fondée sur une mesure de distance inter-chromosomique d et a pour élément de base la fonction appelée *fonction de partage* [17, 1989] . Pratiquement, le fitness d'un chromosome est réduit proportionnellement à l'effectif de ses voisins. La notion de voisinage devient alors très utile et est généralement définie par une fonction d'appartenance

$$Sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha} & si \text{ si } \quad d < \sigma_{share} \\ 0 & \text{sinon} \end{cases}$$

où σ_{share} et α caractérisent respectivement la *largeur* et la *forme* du voisinage. σ_{share} mesure, en quelque sorte, la séparation minimale tolérée entre deux sommets

détectables. L'implantation du partage est alors donnée par le rapport du fitness f sur le cardinal du voisinage c_i :

$$\tilde{f}(i) = f(i)/c_i$$

avec

$$c_i = \sum_{1}^{N} Sh(d_{ij})$$

Les résultats obtenus par Goldberg et Richardson [20, 1987] montrent que l'introduction d'une telle notion améliore la performance des AG. Le partage se stabilise lorsque

$$\forall i, j \in O, \quad \frac{f_i}{c_i} = \frac{f_j}{c_j}$$

où O est l'ensemble des différents optima locaux.

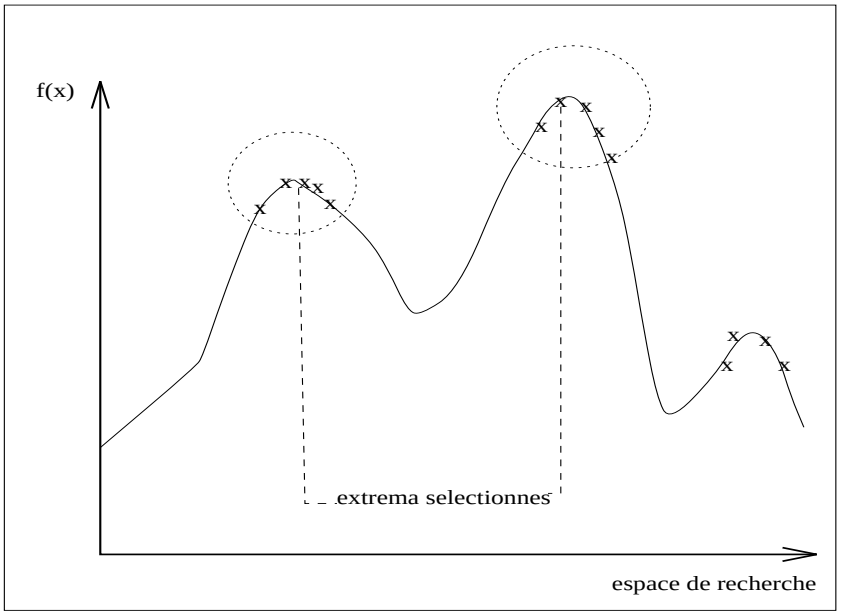


FIG. 10 - *Effet du sharing*

4 Parallélisation massive des AG pour la M.D.S

Nous avons pris conscience de la nécessité impérieuse de paralléliser nos algorithmes dont le temps d'exécution - dès que la taille de l'ensemble à étudier dépasse quelques dizaines - devient vite très lourd; et cela, même si les indices et critères utilisés sont parfaitement adaptés à la nature des données traitées. L'objet de cette partie est de présenter les approches que nous avons adoptées et de mettre en exergue les considérations qui entrent en jeu lorsqu'il s'agit de concevoir des AG dans des environnements multiprocesseurs. Nous commençerons par résumer succinctement la théorie du parallélisme. Ensuite, nous présenterons des approches de parallélisation, orientés environnements multiprocesseurs à mémoire distribuée, des Algorithmes génétiques pour la MDS.

4.1 Expression du Parallélisme

Avant de proposer un algorithme parallèle, il convient de prendre connaissance des outils, des termes et des notions caractéristiques de cette branche de l'informatique que l'on appelle parallélisme [25, 1984], [6, 1985]. Ce chapitre vise très précisément à défricher un champ terminologique par trop exubérant et à présenter des notions préliminaires, en faisant explicitement référence à l'état actuel de l'art. Un détour qui s'avèrera des plus fructueux pour la compréhension de l'intérêt que représente le parallélisme.

4.1.1 Types d'Architectures

La notion de parallélisme évoque une exécution simultanée et judicieusement synchronisée de plusieurs opérations élémentaires. Ce processus est rendu opérationnel au niveau architectural par la disposition de plusieurs processeurs. Les révolutions techniques réalisées dans la conception des circuits intégrés - faible coût de fabrication des dispositifs électroniques tels que les mémoires et les microprocesseurs, miniaturisation des composants électroniques - conduisent au développement d'architectures multi-processeurs assez complexes et très efficaces. Une même méthode de résolution d'un problème donné peut alors conduire à une multitude d'implémentations différentes, suivant l'une des trois grandes approches de parallélisation :

(i) transformation de la solution séquentielle existante en exploitant le parallélisme qu'elle recèle en tenant compte des outils en place.

(ii) adaptation d'une solution parallèle d'un problème similaire.

(iii) invention d'une architecture parallèle adéquate au problème.

Si la première approche est très souvent utilisé dans le domaine de la recherche où l'on dispose la plupart du temps de machines parallèles générales, les industriels préfèrent la troisième approche qui est très matérielle : les machines sont spécialisées donc coûteuses. Un bon nombre de programmes parallèles sont ainsi propres à une architecture donnée. Nous exploiterons dans nos applications l'architecture à topologie variable de la machine Paragon d'Intel.
Dans l'ensemble, une classification des ordinateurs selon la multiplicité des flux d'instructions et de données disponibles matériellement donne lieu à une partition en 3 grandes catégories :

1. SISD; Single Instruction stream Single Data stream [cf. figure 13]. La machine traite un seul flux d'instructions et de données. Exemple : *PC, station de travail*

2. SIMD; Single Instruction stream Multiple Data stream [cf. figure 13]. L'ordinateur exécute un seul flux d'instructions et plusieurs flux de données. Exemple : *Maspar* .

3. MIMD; Multiple Instruction stream Multiple Data stream [cf. figure 11]. La machine exécute plusieurs flux d'instructions et de données. Exemple : *Paragon*

Les SISD, SIMD, MIMD constituent le "trio" des caractéristiques fondamentales des machines parallèles, sur lequel se focalisent la plupart des modèles théoriques et des maquettes de simulation algorithmique. Examinons-les successivement.

L'architecture SISD est l'ordinateur classique de Von Neumann. En plus du traitement séquentiel des instructions, celles-ci peuvent être pipelinées (exécutées en cascade). On classe dans cette catégorie les machines offrant la possibilité d'une exécution pipelinée.
L'architecture SIMD comporte plusieurs unités de traitement (UT) exécutant simultanément les mêmes instructions. Ces unités sont supervisées par une même unité de contrôle - UC -. L'UC diffuse la même instruction à toutes les UT qui opèrent sur des masses de données distinctes issues de flux de données distincts. L'UC commande aussi le réseau synchrone qui établit la connexion entre la mémoire et les processeurs. Cette catégorie englobe les réseaux cellulaires et systoliques.

L'architecture MIMD diffère de la précédente en ce sens qu'une unité de traitement possède sa propre unité de contrôle. Les processeurs peuvent fonctionner

d'une manière totalement asynchrone offrant ainsi une possibilité d'exécution de programmes différents. Une structure MIMD est dite intrinsèque si elle implique des interactions entre différentes unités de traitement. Elle est dite fortement couplée si les interactions entre les UT sont assez importantes. Dans le commerce, la majorité des architectures sont faiblement couplées.

Le mode de programmation SPMD issu de la structure SIMD offre la possibilité d'exécuter un même programme diffusé par l'unité de contrôle sur toutes les unités de traitement. Les données peuvent provenir de flux distincts. On distingue ainsi des architectures SIMD/SPMD et MIMD/SPMD.

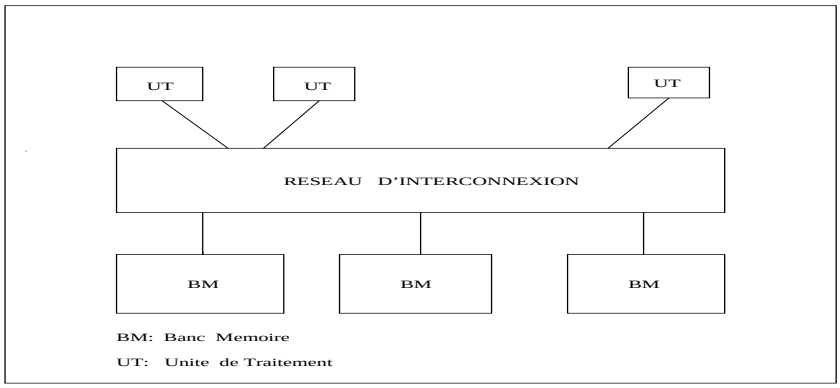


FIG. 11 - *Schéma d'un calculateur SIMD*

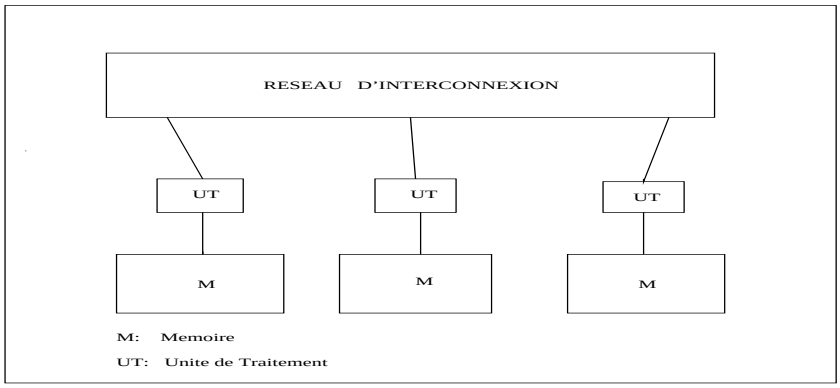


FIG. 12 - *Architecture d'une machine à mémoire distribuée*

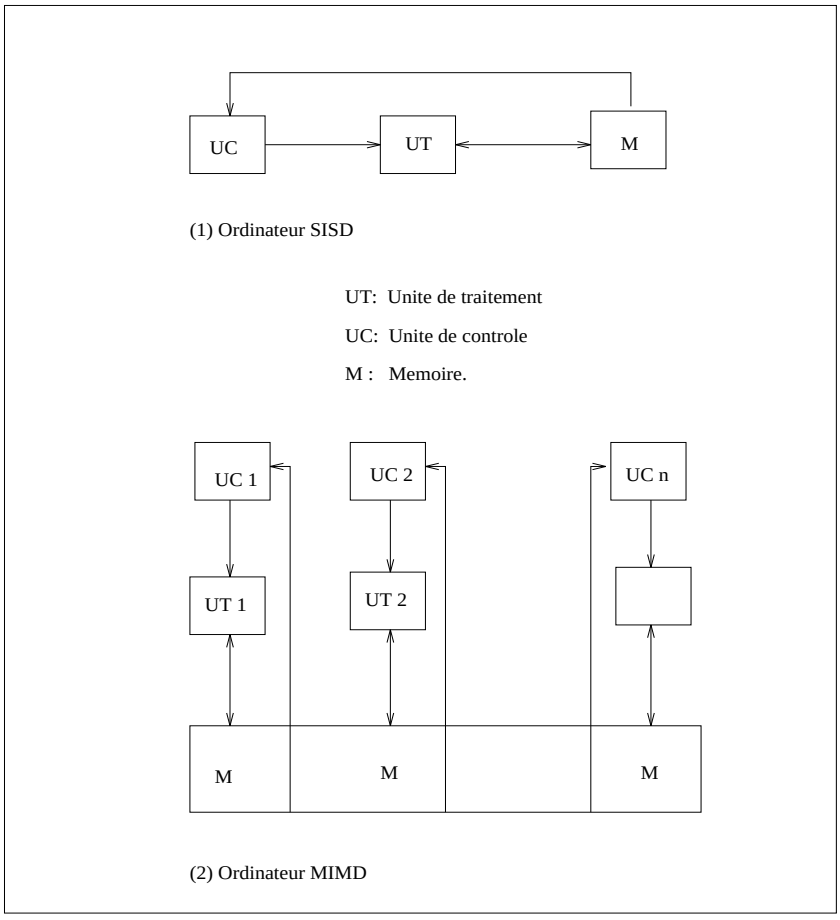


FIG. 13 - *Schémata des calculateurs SISD et MIMD*

4.1.2 Implantation des données

L'une des difficultés fondamentales du traitement parallèle est le mode d'accès aux données. Ce problème devient décisif lorsque l'on souhaite permettre à chaque processeur d'accéder en temps voulu aux données dont il a besoin. Dans les calculateurs vectoriels du type SIMD ou pipe-line, ce problème est de nature combinatoire. En effet, lors de l'exécution d'une instruction nécessitant l'accès à la mémoire centrale, chaque unité de traitement doit lire ou écrire un mot dans cette mémoire. Cette opération conduira fatalement à des conflits d'accès si la mémoire n'est pas divisée en un nombre de modules M au moins égal à celui des unités de traitement

T . Il faut néanmoins garder à l'esprit que la condition $M \geq T$ est nécessaire mais non suffisante pour garantir un accès non conflictuel aux données.
Cette difficulté est étroitement liée à la conception et à la fabrication de la machine parallèle sur laquelle est exécuté l'algorithme. Par conséquent, nous n'aborderons pas les mécanismes informatiques qui permettent de résoudre ce problème d'allocation des données sans conflit.

4.1.3 Tâches, Graphe d'ordonnancement

Pour paralléliser un algorithme, la première étape consiste en la segmentation du programme séquentiel en un ensemble de tâches élémentaires qu'exécuteront les unités de traitement. La partition du problème en sous-tâches et l'élaboration d'un graphe d'ordonnancement permettent, ensuite, de définir les contraintes temporelles pour l'exécution des tâches. L'affectation des tâches aux processeurs tient compte des restrictions liées aux flux d'instructions et d'exécutions, ainsi que des contraintes imposées par la structure de la machine utilisée pour la mise en œuvre de l'algorithme.

On parlera de parallélisme de contrôle lorsque celui-ci porte sur les tâches qui peuvent s'exécuter indépendamment les uns des autres. Lorsque l'on trouve des données qui peuvent être traitées simultanément, le parallélisme est dit de données. Un même algorithme séquentiel peut ainsi conduire à plusieurs versions parallèles, relativement à la synchronisation des processeurs, au mode d'accès des données, au découpage du problème en tâches et à l'affectation de celles-ci aux différentes unités de traitement. Certaines versions seront alors mieux adaptées à une structure d'ordinateur donnée. Une machine SIMD, par exemple, a du mal à traiter un algorithme asynchrone, alors que les problèmes de synchronisation des processeurs pour un ordinateur MIMD sont purement matériels.

4.1.4 Mesure des performances

Considérons un algorithme pour lequel le temps de recherche d'une solution sur un ordinateur parallèle comportant P processeurs identiques est t_p. Si le temps d'exécution de sa version séquentielle - avec un seul processeur - est t_1, on définit deux mesures de qualité:

La performance ou l'accélération (speed-up) d'un algorithme parallèle est le facteur d'accélération donné par le rapport:

$$S_p = \frac{t_1}{t_p}$$

L'efficacité d'un algorithme parallèle correspond au taux moyen d'utilisation des processeurs et le mieux que l'on puisse espérer est d'obtenir $S_p = p$.

$$e_p = \frac{S_p}{p}$$

où p désigne le nombre de processeurs utilisés.

La parallélisation de l'algorithme est d'autant meilleure que l'efficacité est proche du nombre 1 voir supérieure à 1. Dans ce dernier cas, on parle d'une efficacité superlinéaire.
Il convient de garder à l'esprit que les performances d'un algorithme parallèle sont fortement influencées par les contraintes imposées par la structure de la machine utilisée. Le temps de transmission des données varie énormément suivant la topologie adoptée. Notons également que l'on préfère généralement considérer un coût nul pour toute duplication, supposant ainsi qu'il est possible de pipeliner une même donnée vers plusieurs unités de traitement.

4.2 Parallélisation des AG pour la MDS

La parallélisation d'un AG a pour but principal d'augmenter son efficacité en réduisant le temps de calcul. Les différents calculs sont distribués sur un ensemble de processeurs pouvant communiquer via un réseau d'interconnexion (hypercube, réseau maillé,etc). Actuellement, les différentes architectures permettent de classifier l'ensemble des algorithmes génétiques distribués en trois types [15, 1990].

(a) Modèle *parallèle standard* ou modèle *centralisé* [44, 1987]
L'évaluation, le croisement et la mutation se font en parallèle, tandis que la reproduction est séquentielle donc centralisée. Ce modèle n'est pas flexible puisque le coût de communication croît exponentiellement en fonction de l'effectif de la population.

(b) Modèle de migration
La population est divisée en sous populations de même taille. Chaque processeur exécute l'algorithme standard sur la sous population qui lui est affectée.

Pour propager les meilleurs individus entre les sous population, les noeuds échangent périodiquement des données. Les meilleurs individus reçus vont ainsi remplacer les plus mauvais de la population locale. Cet échange empêche chaque noeud de tomber dans un optimum local. La fréquence des échanges et le nombre des individus affectés sont des paramètres de l'algorithme. Dans cette approche, le parallélisme inhérent aux algorithmes génétiques n'est pas totalement exploité, le traitement d'une sous population étant séquentielle. Ce modèle est cependant intéressant dès lors que le nombre de processeurs disponibles est plus petit que la taille de la population désirée. De plus, on peut éviter le coût et la complexité des algorithmes de routage dans les architectures parallèles à mémoire distribuée en restreignant les échanges d'individus aux processeurs directement connectés.

(c) Modèle de *diffusion*

Chaque processeur traite un seul individu et toutes les opérations se font en parallèle. East et Macfarlane [15, 1990] ont montré que cette approche donne en général les meilleures solutions avec un minimum de temps dans différents schémata d'exécution.

La suite de cette partie est consacrée à la présentation de plusieurs algorithmes qui se prêtent à une bonne parallélisation de nos algorithmes génétiques. Le support d'expérimentation de nos algorithmes est la Paragon d'Intel qui est doté d'une architecture multiprocesseur à mémoire distribuée [34, 1994].

4.2.1 La machine PARAGON XP/S i860 d'Intel

La machine Paragon est un super-ordinateur MIMD d'Intel appartenant à la classe des machines massivement parallèles à mémoire distribuée [cf. figure 12]. Une autre classe très consistante d'ordinateurs parallèles est celle des machines massivement parallèles à mémoire partagée.
La structure d'une machine dite à mémoire distribuée se présente sous forme d'un ensemble de nœuds identiques reliés par un réseau d'interconnexion. Les machines de ce type se distinguent essentiellement par leur module opératoire qui est en réalité un ordinateur indépendant avec sa mémoire et ses processeurs. Notons également la spécificité de la topologie de leur réseau d'interconnexion quelquefois dynamiquement reconfigurable. On distingue généralement quatre types de topologie : hypercube,

tore, linéaire et grille. Une conséquence immédiate de l'absence d'une mémoire commune est que les processus ne peuvent interagir qu'en s'échangeant des messages routés par le réseau d'interconnexion.

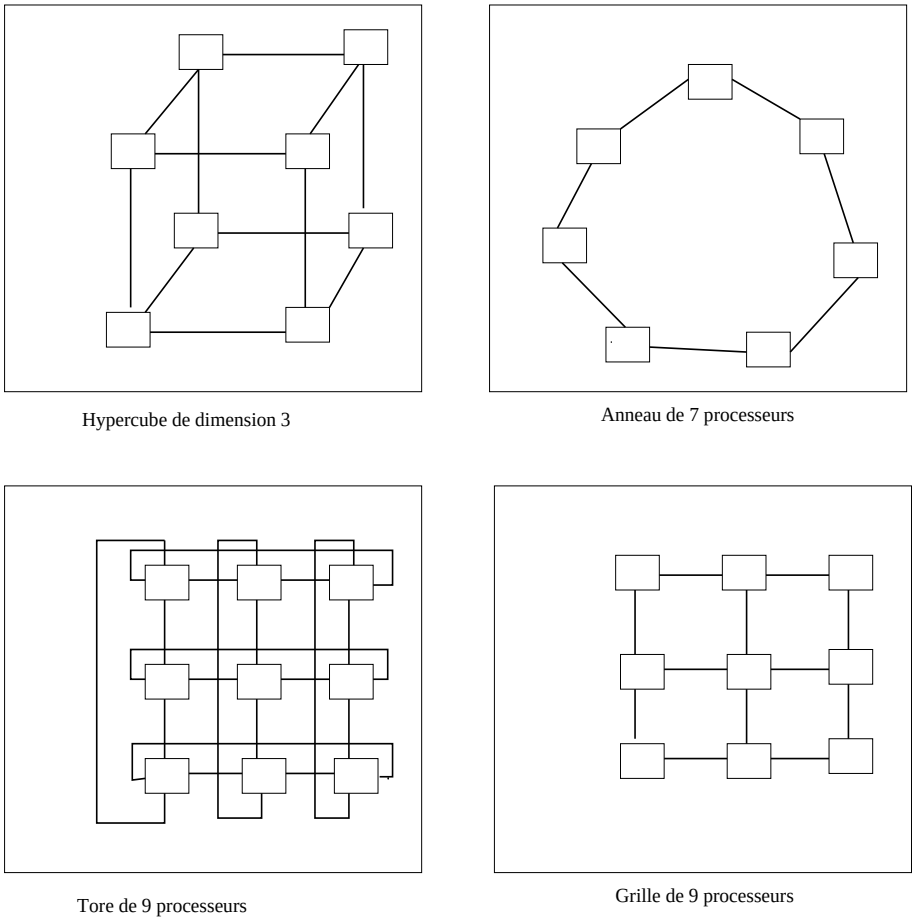


FIG. 14 - *Configurations utilisées pour la parallélisation*

Le calculateur Paragon de l'IRISA [cf. figure 15], qui peut évoluer, est doté actuellement d'une soixantaine de nœuds qu'on peut diviser en deux parties :

♠ l'unité de gestion constituée de six nœuds systèmes ou de service;

♠ l'unité de calcul composée de 56 nœuds de traitement.

Un nœud de calcul comprend une mémoire de $16Mo$ extensible et deux processeurs de type i860 dont l'un s'occupe essentiellement de la gestion des communications et du routage des messages.

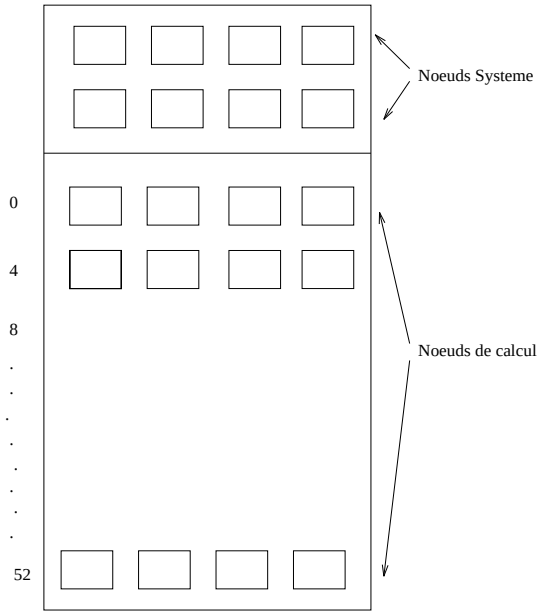


FIG. 15 - *Configuration du calculateur Paragon*

4.2.2 Mise en oeuvre des algorithmes

Les algorithmes génétiques sont fondamentalement parallèles. En fait, l'essentiel des calculs est effectué dans une boucle **répéter ... jusqu'à** . On evalue dans chaque boucle, un ensemble d'opérations identiques. La mise en oeuvre parallèle consiste à faire ces opérations par plusieurs processeurs. Grâce au réseau d'interconnexion de la Paragon qui garantit un faible coût de communication, notre première approche à été de centraliser la reproduction-sélection. Cette approche parallélise précisément les évaluations du fitness car ce sont elles qui constituent l'essentiel du calcul. La stratégie dans cet algorithme consiste à distribuer, de façon équitable, les individus de la population traitée aux processeurs qui se charge d'évaluer leur fitness. Dans

Systeme

	OSF	SUNMOS
Start-up	93 us	50 us
Bandwidth	30 Mo/s	120 Mo/s

Processeur i860

Frequence	50 MHz
Double precision peak	75 MFlops
Single precision peak	100 MFlops

FIG. 16 - *Caractéristiques du système et du processeur i860*

ces conditions, l'algorithme orienté environnements multiprocesseurs à mémoire distribuée et inteconnecté (physiquement ou logiquement) se résume en trois étapes qui s'ajoutent au prétraitement usuel :

1. Le processeur P_0 envoie à chaque unité de traitement la matrice de dissimilarité. P_0 distribue également aux autres processeurs un nombre de configurations proportionnel à l'effectif total des nœuds mis à contribution.

2. Chaque nœud évalue les configurations qu'il a reçues, puis envoie au processeur P_0 le vecteur coût local correspondant aux résultats de ses différentes évaluations.

3. La dernière étape consiste à construire de nouvelles configurations par des opérateurs génétiques. Pour cela, P_0 s'occupe de choisir les configurations à croiser d'une part, les configurations dont quelques éléments subiront des mutations d'autre part, et le reste de l'opération de croisement ou de mutation se poursuit au niveau des différents nœuds.

Le mode de reproduction standard introduit, dans cette alternative, une forte synchronisation car l'opération est faite après que tous les individus aient été évalués. Nous avons donc orienté nos recherches vers une décentralisation de l'opération de reproduction. En particulier, nous avons implanter une version parallèle où chaque nœud exécute l'algorithme génétique séquentiel sur une sous population. Un échange des meilleurs individus entre les sous populations se fait régulièrement. Ainsi, les processeurs ne risque pas de tomber, d'une façon prématurée, dans un optimum local.

5 Tests numériques

Nous présentons, dans ce paragraphe, les résultats se rapportant aux codes Fortran[11] double précision des implémentations séquentielles et parallèles sur des données réelles. Pour les performances de l'algorithme parallèle évoquées ci-dessous, les populations initiales sont construites de façon aléatoire et la topologie considérée est l'anneau. Par ailleurs chaque processeur effectue toujours une seule migration. Ainsi, les sous-populations peuvent rester très différentes sans que le coût des communications ne devienne un handicap.

L'algorithme à été testé sur un certain nombre de problèmes classiques dont on sait apprécier les solutions optimales. Pour la mesure des performances des implémentations, les temps d'exécution reportés correspondent à 250 itérations de l'AG, soit 16000 évaluations du fitness. La population initiale contient 64 individus. Ce chiffre correspond au p.p.m.c des différentes tailles d'anneaux considérés. Les probabilités d'utilisation de l'opérateur de mutation sont de 10% tandis que celles de l'opérateur de croisement - global ou local - est de 80%. Ces différents paramètres de simulation sont fixés conformément aux valeurs préconisées par la théorie des AG et nous sommes, à cette date, incapables de proposer des valeurs optimales.

La figure 17 montre le speed-up obtenu dans une simulation ayant les caractéristiques décrites ci-dessus et portant sur une matrice de dissimilarité de taille 10, soit 20 variables pour le fitness. Precisons que cette simulation est independante du caractère affine de la matrice utilisée. Elle depend uniquement de la taille de la matrice et du degré du parallélisme du programme. L'arrêt du programme est guidé uniquement par le nombre d'itérations maximal fixe. Le stress est donc mis

11. version 77

Number of processors	Total CPUtime per 200 generations
1	0.3219 E+02
4	0.8942 E+01
8	0.4369 E+01
16	0.2136 E+01
32	0.1523 E+01

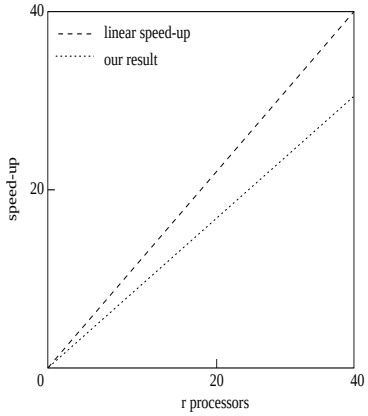


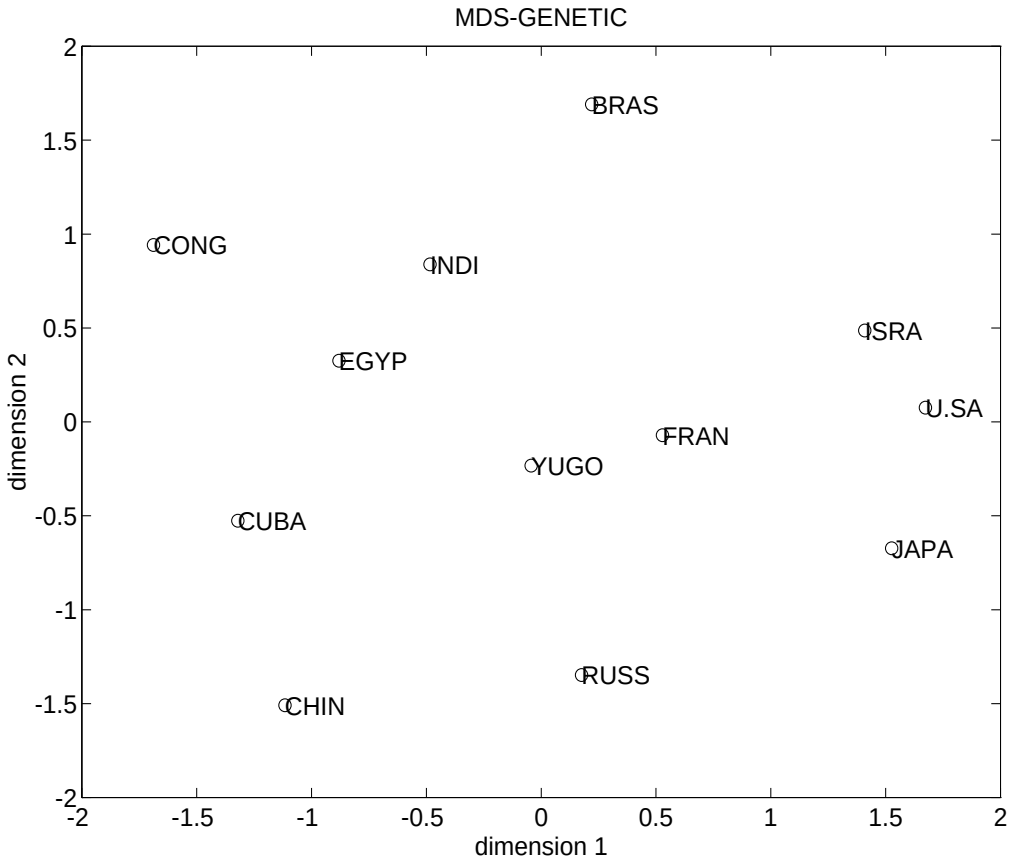
FIG. 17 - *Timing en secondes et speed-up par rapport à un processeur*

entre parenthèse. Comme l'on pouvait s'y attendre, l'algorithme possède un speed-up presque linéaire[12]. Ceci s'explique par le fait que le coût de communication entre processus - compte tenu de l'efficacité du réseau d'interconnexion de la machine paragon - est relativement petit comparé au coût d'exécution des processus.

Lors de nos expériences nous avons dû faire face à des problèmes de dépassement de capacité de mémoire vive lorsque nous exécutons de gros problèmes avec l'AG parallèle. L'insuffisance de place mémoire RAM cause des défauts de page et les mécanismes de swap faussent complètement les mesures.

Dans la phase d'application aux données réelles, le premier tableau de données est une matrice de proximité contruite sur un ensemble de 12 pays et tirée de [28, 1984]. Les données sont issues d'une étude pilote sur la perception des nations. Chacun des 18 étudiants participant à l'étude attribue une note entre 1 - pour très différent - et 9 - pour très similaire - à chacun des 66 paires formées dans l'ensemble des nations considérées. Après toutes les normalisations, on constate sur la matrice que la Russie et la Yougoslavie[13] présentent la plus grande similarité, alors que les couples Chine-Brésil et U.S.A-Congo ont été jugés comme présentant le plus de dissemblances. Le résultat - STRESS = 0.0013 - obtenu par l'AG au bout de 500 itérations avec une configuration aléatoire de taille 100 est le suivant:

12. Les données de petite taille conduisent à un speed-up superlinéaire.
13. Ces données datent de la période communiste.



TAB. 2 - *Représentation de 12 pays capitalistes et communistes*

Une interprétation de cette configuration[14] montre un positionnement très intérressant des pays sur le plan: Les U.S.A, le Japon, la France et Israël qui sont des pays capitalistes dévéloppés. Le Congo, l'Egypte et l'Inde qui sont des pays sous-dévéloppés. Le Cuba, la Chine, la Russie et la Yougoslavie qui sont des pays communistes. Cette interprétation plutôt d'ordre économique nous paraît bien plausible et n'exclut en aucun cas d'autres interprétations.

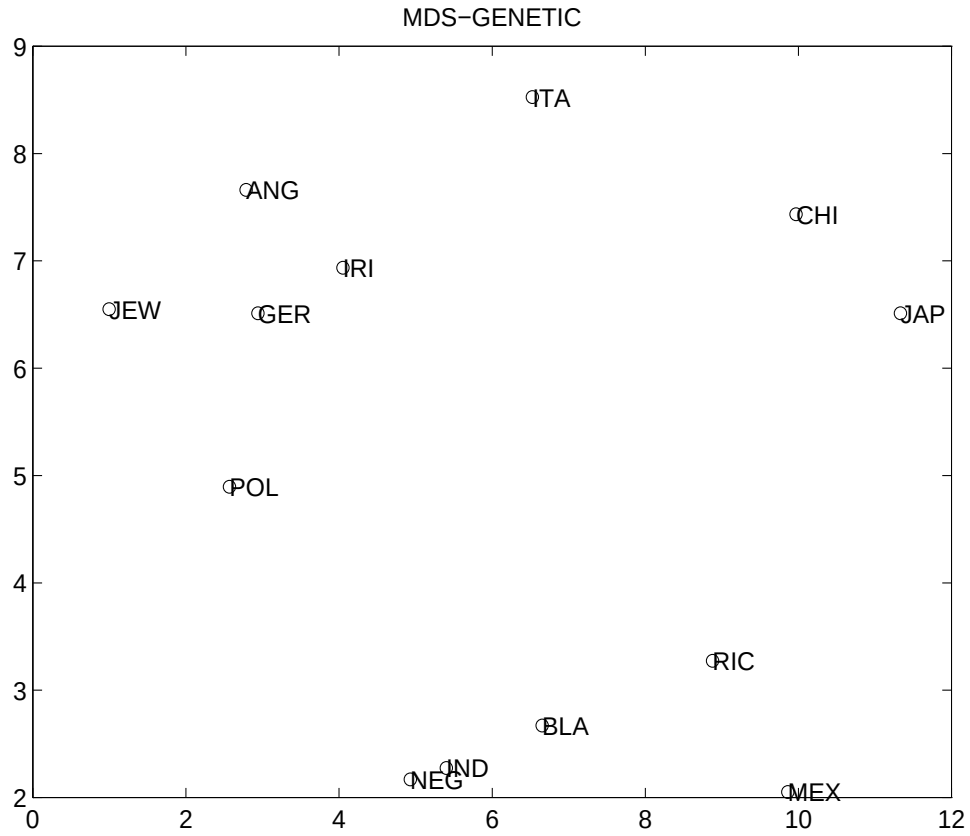
Un second jeu de données qui nous permet d'établir une comparaison de notre algorithme avec les algorithmes itératifs de recherche locale est tiré de [23, 1986].

14. La signification des différents symboles est donnée en annexe.

Il s'agit d'une matrice de dissimilarité plus classique portant sur un groupe de 13 grandes ethnies, nations ou peuples. La matrice utilisée dans cette seconde analyse a l'avantage d'être très peu euclidienne et d'avoir été au centre d'une confrontation entre les deux logiciels les plus performants du marché. La figure 18 montre l'amélioration obtenue par les AG. Notons cependant que la convergence n'est pas toujours régulière mais est quelquefois sujette à des oscillations. C'est pour cette raison que nous conseillons fortement l'utilisation des versions hybrides des AG. Numériquement, une version hybride des AG donne dans le pire des cas des résultats aussi bons que ceux obtenus par un algorithme de descente. Un test de l'AG hybride - gradient classique - avec la matrice de données précédente produit la configuration de la figure TAB-3 avec un STRESS de 0.0991 en 45.15 secondes. Ce qui nous semble très satisfaisant.

Methods	Stress values	Iterations	Initial Configuration
ALSCAL	0.31259 E+00	5	Torgerson
SMACOF	0.24550 E+00	32	Torgerson
GA (serial)	0.13181 E+00	100	Random

FIG. 18 - *Confrontation des AG. aux autres méthodes.*



TAB. 3 - *Représentation de 13 grands groupes ethniques*

6 Conclusion

Les aspects du cacul parallèle de la représentation affine d'une matrice de proximité par les algorithmes génétiques ont été proposés et évalués en détail dans ce rapport. Nous avons étudié

(i) les méthodes numériques et

(ii) comment les implanter dans les environnements parallèles.

Nous avons démontré la faisabilité de l'utilisation des AG au coeur même de la MDS et proposé des approches d'hybridation. Par ailleurs, des tests numériques effectués produisent des résultats très encourageants. Les comparaisons sont faites avec les deux logiciels - à transitions déterministes - les plus performants du marché.

Dans la démarche que nous avons entreprise, les algorithmes sont orientés architectures multiprocesseurs à mémoire distribuée. Cependant, les versions alternatives pour les machines à mémoire partagée peuvent être facilement générées à partir de la stratégie de parallélisation que nous avons proposée.

Les algorithmes parallèles possède un speed-up presque linéaire. Toutefois, les routines que nous avons mises en oeuvre dans les versions parallèles affichent un mauvais comportement lorsque le surcoût des communications - dû à des matrices de dissimilarité de grande taille - est élévé. Plusieurs directions seront prospectées pour améliorer leur efficacité. Par exemple, au lieu de considérer un anneau, on pourrait adopter des topologies 2D plus flexibles. Sans doute la direction la plus prometteuse passera par une décomposition additive de la fonction coût qui minimisera - c'est notre souhait - les transferts inter-processeurs.

Une autre étude d'amélioration de nos algorithmes portera sur la variation dynamique des paramètres de l'algorithme et particulièrement celle de la probabilité de mutation et la fréquence d'intervention des algorithmes de descente déterministes dans les versions hybrides. Enfin, il est intéressant, notamment pour une diversité effective des individus initiaux, de pouvoir injecter,"si cela ne coûte pas cher", dans la population initiale, des chromosomes qui sont relativement de bonne qualité. La méthode des "pôles d'attraction" [32, 1977], mise au point et développée dans les années 75, répond parfaitement à cet objectif.

Remerciements

Nous tenons à remercier Pascal Lemonnier de l'équipe API (IRISA) pour ses conseils sur les tournures du parallélisme, Gilbert Cabillic de l'équipe SOLIDOR (IRISA) et Hugue Leroy (IRISA) pour leurs conseils sur l'utilisation de la Paragon.

Références

[1] F. Baiardi and S. Orlando. Strategies for a massively parallel implementation of simulated annealing. In Netherlands LNCS, Eindhoven, editor, *PARLE'89*, pages 273–287, June 1989.

[2] A. Bertoni and M. Dorigo. *Implicit parallelism in genetic algorithm.* Technical Report 92-012, Politecnico di Milano, 1992.

[3] C. L. Bridges and D. E. Goldberg. The nonunifom walsh-schema transform. In *FOGA91*, pages 13–22, 1991.

[4] F. Cailliez and J. P. Pages. *Introduction à l'Analyse des données.* SMASH, 1976.

[5] R. Cerf. *Une théorie asymptotique des algorithmes génétiques.* PhD thesis, Université de Montpellier 2, March 1994.

[6] M. Cosnard and Y. Robert. *Algorithme parallèle pour machines SIMD et MIMD.* RR 554, IMAG, September 1985.

[7] G. D. d' Aubigny. *l'Analyse Multidimensionnelle Des Données De Dissimilarité.* PhD thesis, Universite Joseph Fourier- Grenoble I, January 1989.

[8] T. Starkweather D. Whitley and C. Bogard. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14(3):347–361, August 1990.

[9] G. d'Aubigny. Analyse des proximités et programmes de codage multidimensionnel. *Revue de Modulad*, 12:1–32, July 1994.

[10] Y. Davidor. Genetic algorithms and robotics. an heuristic strategy for optimization. *World Scientific*, 1, 1990.

[11] L. Davis. *Genetic algorithms and simulated annealing.* London : Pitman ; Los Altos, CA : Morgan Kaufmann, 1987.

[12] T. E. Davis and J. C. Principe. A simulated annealing like convergence theory for simple genetic algorithm. *Proceedings of the Fourth International Conference on Genetic Algorithm.*, 174–182, July 1991.

[13] J. de Leeuw and J. Meulman. A special jackknife for multidimentional scaling. *Journal of Classification*, 3:97–112, 1986.

[14] K. Deb and D. E. Goldberg. *Analysing deception in trap functions.* Technical Report, University of Illinois at Urbana-Champaign, December 1991.

[15] I. East and D. Macfarlane. An investigation of several parallel genetic algorithms. *University of Bucckingham*, 60–67, April 1990.

[16] D. E. Golberg and P. Segrest. Finite markov chain analysis of genetic algorithms. *Proceedings of the Second International Conference on Genetic Algorithms*, 1–8, 1987.

[17] D. Goldberg. *Genetic Algorithms in Searh, Optimization and Machine Learning.* Addison-Wesley, 1989.

[18] D. E. Goldberg. *Computer-aided gas pipeline operation using genetic algorithms and rule learning.* PhD thesis, University of Michigan., 1983.

[19] D. E. Goldberg. *Construction of high-order deceptive functions using low-order walsh coefficients.* Technical Report 90002, University Illinois at Urbana-Champaign, December 1990.

[20] D. E. Goldberg and J. Richardson. *Genetic algorithms with sharing for multidimodal function optimization.*, pages 41–49. Hillsdale, New Jersey,, 1987.

[21] R. Gunter. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on neural network*, 5(1):96–101, January 1994.

[22] K. Deb H. Kargupta and D. E. Goldberg. *Ordering genetic algorithms and deception.* Technical Report 92006, University of Illinois-Champaign at Urbana-Champaign, April 1992.

[23] W. Heiser. Smacof-1. *University of Leiden*, 1986.

[24] J. H. Holland. *Adaptation in Naturel and Artificial Systems.* PhD thesis, University of Michigan, 1975.

[25] K. Hwang and F. A. Briggs. *Computer architecture and parallel processing.* McGraw-Hill, 1984.

[26] M. Iosifescu. *Finite Markov Processes and Their Applications.* Chichester: Wiley, 1980.

[27] J. J. Gefenstette J. M. Fitzpatrick and D. Van Gucht. Image registration by genetic search. *In Proceedings of IEEE Southeast conference.*, 1984.

[28] J. B. Kruskal. Nonmetric multidimensional scaling. *Bell Telephone system*, 4821, June 1964.

[29] J. B. Kruskal and M. Wish. *Multidimensional Scaling.* Sage Publications, 1984.

[30] H. Leredde. *La Méthode des Pôles d'attraction; la méthode des pôles d'aggrégation: deux nouvelles familles d'algorithmes en Classification automatique et sériation.* PhD thesis, Universite de Paris VI, October 1979.

[31] I. C. Lerman. *Nombre de solutions et satisfiabilité d'un problème SAT; Une approche ensembliste, combinatoire et statistique.* Technical Report 600, IRISA, September 1991.

[32] I. C. Lerman and H. Leredde. La méthode des pôles d'attraction. In Versailles IRIA, editor, *Journées Analyse des Données et Informatique*, 1977.

[33] I. C. Lerman and R. F. Ngouenet. Un rôle pour les algorithmes génétiques dans la représentation euclidienne d'une matrice de dissimilarité. In Yadolah Dodge, editor, *XXVI ème journées de Statistique*, pages 426–429, April 1994.

[34] H. Leroy. *Intel Paragon: Architecture, Programmation et Outils.* Technical Report, IRISA, November 1994.

[35] E. Lutton. *Etat de l'art des Algorithmes Génétiques.* Technical Report, INRIA-Rocquencourt, February 1994.

[36] M. Machmouchi. *Contributions à la mise en oeuvre des méthodes d'Analyse des Données de Dissimilarité.* PhD thesis, Université Pierre Mendes-France Grenoble II, October 1992.

[37] A. Mead. Review of the development of multidimensional scaling methods. *The Statistician*, 41:27–39, 1992.

[38] M. Nei. Molecular evolutionary genetics. *Columbia University Press, New York*, 1987.

[39] R. F. Ngouenet. *Une nouvelle famille d'indices de dissimilarité pour la MDS.* Technical Report 766, IRISA, Universite de Rennes-1, October 1993.

[40] S. S. Rao. *Optimization, Theory and Applications.* Wiley, 1984.

[41] G. Robertson. Parallel implementation of genetic algorithms in classifier system. *Genetic algorithms and simulated annealing*, 129–140, 1987.

[42] M. Reynolds S. Schiffman and W. Young. Introduction to multidimensional scaling, theory, methods and applications. *Academic Press*, 1981.

[43] J. E. Savage and M. G. Wloka. On parallelizing graph-partitioning heuristics. *Automata, Languages and Programming.*, (443):476–489, July 1987.

[44] J. Y. Suh and D. V. Gucht. *Distributed genetic algorithms.* Technical Report 225, Computer science departement, Indiana University, July 1987.

[45] Schaffer Voelker and Mukai. Sponteneous allozyme mutations in drosophilia melanogaster : rate of occurence and nature of the mutants. *Genetics 94*, 961–968, 1980.

Annexe

TAB-2	TAB-3
1-BRA BRASIL	1- ANG ANGLO
2- CONG CONGO	2- BLA BLACK
3- CUBA CUBA	3- CHI CHINESE
4- EGYP EGYPTE	4- GER GERMAN
5- FRAN FRANCE	5- IND INDIAN
6-INDI INDIA	6- IRI IRISH
7- ISRA ISRAEL	7-ITA ITALIAN
8- JAPA JAPAN	8-JAP JAPANESE
9- CHIN CHINA	9- JEW JEWISH
10- RUSS RUSSIA	10- MEX MEXICAN
11- U.SA U.S.A	11- NEG NEGRO
12- YUGO YUGOSLAVIA	12- POL POLISH
	13- RIC PUERTO-RICAN

Symboles et noms des entités étudiées dans le rapport.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399