UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM
FACULTÉ DES SCIENCES EXACTES ET INFORMATIQUE
DÉPARTEMENT DE MATHÉMATIQUES

Mémoire de fin d'étude

Pour l'obtention du diplôme de Master de Mathématiques

Cycle LMD

Spécialité : Modélisation Contrôle et Optimisation

Thème :

Algorithme Génétique

Présenté par :

MEHIDID Fadila

Soutenu le 19/06/2013.

Les membres de jury

Président	MCA	Belhamiti	Omar	**U. MOSTAGANEM.**
Examinateur	MCA	AMIR	Abdessamad	**U. MOSTAGANEM.**
Encadreur	MAA	ABLAOUI	Hocine	**U. MOSTAGANEM.**

Table des matières

Remerciments

Introduction Général

L'algorithme génétique est une méthode inspirée par la théorie de l'évolution telle qu'elle a été définie par le naturaliste britannique Charles Darwin au siècle dernier dans son livre intitulé «The Origin Of Spacies» [5] .Dans ce livre, Darwin constata que l'évolution des espèces est basée sur deux composantes : la sélection et la reproduction. La sélection garantit une reproduction plus fréquente des chromosomes des êtres vivants les plus robustes, tandis que la reproduction est une phase durant laquelle l'évolution se réalisera et les descendants obtenus ne sont pas reproduits à l'identique. Et c'est John Holland dans ces travaux sur les systèmes évolutifs, en 1975, trouver le premier modèle formel des algorithmes génétiques (the canonical genetic algorithm AGC) dans son livre « Adaptation in Natural and Artificial Systems ». Il expliqua comment ajouter de l'intelligence dans un programme informatique avec les croisements (échangeant le matériel génétique) et la mutation (source de la diversité génétique). Ce modèle servira de base aux recherches ultérieures et sera plus particulièrement repris par Goldberg, un ouvrage de vulgarisation des algorithmes génétiques à travers son livre "Genetic Algorithms in Search, Optimization, and Machine Learning " (1989). Ce livre est encore édité aujourd'hui et reste le "best-seller" des algorithmes génétiques.

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastiques, métaheuristiques, appartenant à la famille des algorithmes évolutionniste, fondés sur les mécanismes de la sélection naturelle et de la génétique, d'où ils ont pris leur nom «algorithme génétiques». Cette dernière méthode a connu un regain d'intérêt ces dernières années grâce sa souplesse de traiter des fonctions objectif non régulière ou des fonctions définies sur des espaces de recherche non standard (espaces discrets,. . . etc.). et de leurs champs d'applications dans plusieurs domaines : optimisation de fonctions numériques difficiles, traitement d'image, optimisation d'emplois du temps, contrôle de systèmes industriels [Beasley, 1993a], cryptographie, apprentissage des réseaux de neurones [Renders, 1995], finance [Pereira , 2000],et encore en théorie des jeux par Axelrod . . . etc.

Ce mémoire est divisé en quatre chapitres :

Le premier chapitre présente une description détaillée des algorithmes génétiques dans lequel nous donnons quelques définitions. Le deuxième chapitre sera consacré aux aspects théoriques de l'algorithme génétique, au troisième chapitre nous donnons une mise en œuvre des différents types d'opérateurs Génétiques sur un exemple. Enfin, le dernier chapitre est consacré à l'application de l'algorithme génétique aux problème du P-médian.

Chapitre 1

Algorithme génétique

1.1 Introduction

Les algorithmes génétiques constituent une approche originale : il ne s'agit pas de trouver une solution analytique exacte ou une bonne approximation numérique, mais de trouver des solutions satisfaisant au mieux à différents critères, souvent contradictoires. S'ils ne permettent pas de trouver à coup sûr la solution optimale de l'espace de recherche, du moins peut-on constater que les solutions fournies sont généralement meilleures que celles obtenues par des méthodes classiques.

Comme nous l'avons développé dans l'introduction, les origines des AG sont assez intuitives. Mais nous allons voir qu'ils impliquent des aspects assez techniques. En effet, dans ce chapitre, nous avons établi les fondements nécessaires à la compréhension des algorithmes génétiques. Tout d'abord, nous présentons L'aspect biologie dans AG. La partie suivante constitue les principes généraux d'algorithme génétique, ensuite on doit exposer en détail les différentes étapes qui constituent la structure générale d'un algorithme génétique : Codage, méthode de sélection, opérateurs de croisement et de mutation avec leurs probabilités, méthode d'insertion et les tests d'arrêt. Enfin, nous terminons par une conclusion.

1.2 Terminologie

Avant d'aborder comment fonctionnent les AGs, nous devons également définir la terminologie employée :

Chromosome : En biologie, il est défini comme le porteur de l'information génétique nécessaire à la construction et au fonctionnement d'un organisme. Dans le cadre des AG, il correspond à un élément représentant une solution possible d'un problème donné.

Gène : En biologie, il représente une partie du chromosome, chaque chromosome est constitué d'un certain nombre de gènes. Pour un AG, chaque chromosome est divisé en un ensemble d'unités le constituant *appelé* gènes.

Génotype : Dans les systèmes naturels, l'ensemble du "matériel" génétique est appelé le génotype. Dans les AG, l'ensemble des chaînes est appelé structure.

Phénotype : Dans les systèmes naturels, l'organisme formé par l'interaction de l'ensemble du "matériel" génétique avec son environnement est appelé phénotype. Dans les AG, les structures décodées forment un ensemble de paramètres donnés, ou solutions ou bien points de l'espace des solutions.

Allèle : Dans les systèmes naturels, l'allèle est une composante du gène. Les allèles sont les différentes valeurs que peuvent prendre les gènes. Dans les AG, l'allèle est également appelé valeur caractéristique.

Locus : Le locus est la position d'un gène dans le chromosome.

Individu : En biologie un individu est une forme qui est le produit de l'activité des gènes. Pour un AG, il est réduit à un chromosome et on l'appelle donc chromosome ou individu pour désigner un même objet.

Population : Dans un système naturel, une population est simplement un ensemble d'individus. Par analogie, elle se définit comme l'ensemble des chromosomes. Elle est aussi appelée une génération.

Parents : Dans un système naturel, les individus peuvent se reproduire en créant de nouveaux individus formant une nouvelle génération afin d'assurer la continuité de la vie.

Dans le cadre d'un AG, les parents correspondent aux individus pouvant s'imposer pour donner naissance à de nouveaux individus descendants afin de former une nouvelle génération.

1.3 Principe de fonctionnement des AGs

Le fonctionnement de tout AG peut être décrit par le principe illustré par *la figure* 1.1

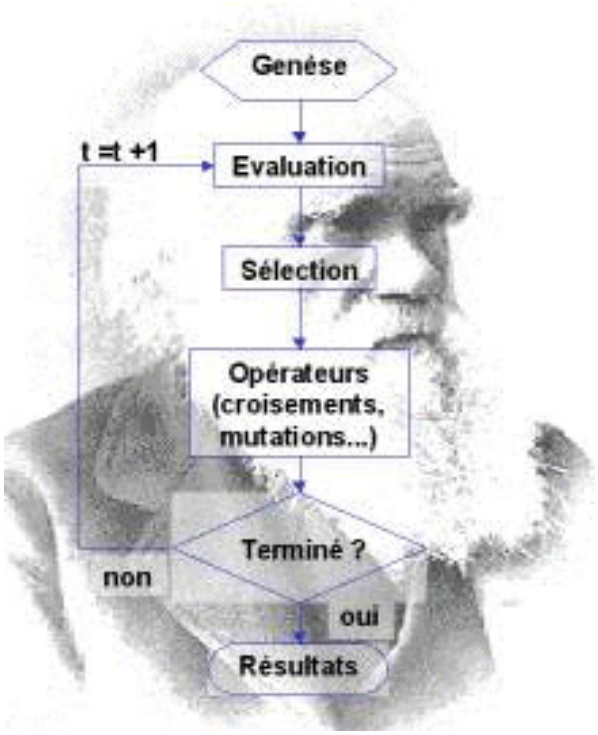


Figure 1.1 *Principe de fonctionnement d'un AG*

En règle générale, pour pouvoir exploiter efficacement le potentiel d'un AG, indépendamment du problème traité, il est demandé de prendre en compte les principes fondamentaux suivants :

1. Un principe de codage des individus de la population : cette étape se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès de l'algorithme génétique.
2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de reproduire une population d'individus qui servira de base pour les générations futures. Le choix de la population initiale est important car il influe sur la rapidité de la convergence vers un optimum global.

3. Définir une fonction d'évaluation appelée généralement « fitness ». Cette dernière a pour objectif d'évaluer une solution et la comparer aux autres ;
4. Choisir les solutions par un mécanisme de sélection pour un éventuel couplage.
5. Générer de nouvelles solutions à l'aide des opérateurs de croisement et de mutation. En effet, l'opérateur de croisement recompose les gènes d'individus existant dans la population, alors que l'opérateur de mutation a pour but de garantir l'exploration de l'espace de recherche en introduisant des nouveaux individus.
6. Etablir un compromis entre les solutions produites (progénitures) et les solutions productrices (les parents) en utilisant un mécanisme d'insertion.
7. Un test d'arrêt qui joue un rôle primordial dans le jugement de la qualité des individus.

1.4 Les caractéristiques des AGs

Les algorithmes génétiques se caractérisent par quatre aspects : le codage des paramètres du problème, l'espace de recherche, la fonction d'évaluation servant à sélectionner les chromosomes parents, et le hasard qui joue un rôle important dans l'évolution des chromosomes de génération en génération. Nous allons passer en revue ces différents aspects.

1.4.1 Le Codage

Premièrement, il faut représenter les différents états possibles de la variable dont on cherche la valeur optimale sous une forme utilisable par un AG : c'est le codage. Cela permet d'établir un lien entre les valeurs de la variable et les individus de la population. Le codage dépend de la spécificité du problème et conditionne fortement l'efficacité de l'algorithme. Il existe trois types de codage :

Le codage Numérique : si "l'alphabet" est constitué des chiffres.

Le codage Symbolique : si "l'alphabet" est un ensemble de lettres alphabétiques ou des symboliques.

Le codage Alph-numérique : si nous utilisons un alphabet combinant les lettres et les chiffres.

Dans le type numérique il y a 2 méthodes : Le codage binaire (représentation sous forme de chaîne binaire) et le codage réel (représentation directe des valeurs réelles de la variable). Nous pouvons facilement passer d'un codage à l'autre (voir Michalewicz (1992))[11].

Le codage réel :

Il a le mérite d'être simple. Chaque chromosome est en fait un vecteur dont les composantes sont les paramètres du processus d'optimisation. Par exemple, si on recherche l'optimum d'une fonction de n variables $f(x_1, x_2, ..., x_n)$, on peut utiliser tout simplement un chromosome ch contenant les n variables :

ch :

x_1	x_2	...	x_{n-1}	x_n

Avec ce type de codage, la procédure d'évaluation des chromosomes est plus rapide vu l'absence de l'étape de transcodage (du binaire vers le réel). Les résultats donnés par Michalewicz (Michalewicz, 1992) montrent que la représentation réelle aboutit souvent à une meilleure précision et un gain important en termes de temps d'exécution. Néanmoins ce codage présent l'inconvénient majeure du la non réalisabilité (souvent) des solutions obtenues après application de différents types d'opérateurs.

Le codage binaire :

Ce codage a été le premier à être utilisé dans le domaine des AG, Il présente plusieurs avantages : les opérateurs de croisement et de mutation sont souvent très simples à manipuler. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus. Néanmoins ce type de codage présente quelques inconvénients : l'utilisation d'un tel codage influe négativement sur la vitesse de convergence de l'algorithme génétique appliqué à des problèmes d'optimisation de grand taille (Michalewicz, 1992)*[11]*. Pour mesurer la dissimilarité entre deux solutions quelconques (codées en binaire), la distance de Hamming est utilisée. Cette dernière présente l'inconvénient que la distance de Hamming entre deux solutions voisines peut être très grande. Exemple les entiers 7 et 8, deux entiers successifs donc très proches, qui correspondent, respectivement, aux chaînes 0111 et 1000 ont une distance de Hamming égale à 4. Ce phénomène peut être la cause du la non convergence de l'algorithme vers l'optimum global.

Malgré toutes les critiques qui lui ont été adressées, le codage binaire reste le plus utilisé.

Définition 1.4.1 *Séquence (Codage binaire)*

Nous appelons (chaîne, chromosome, individu) A de longueur $l(A)$ une séquence

$$A = \{a_1, a_2, ..., a_l\} \ avec \forall i \in \{1, ..., l\}, \ a_i \in V = \{0, 1\}. \tag{1.4.1}$$

Un chromosome est donc une suite de bits (formée de zéros et de uns), appelé aussi chaîne binaire.

Définition 1.4.2 *[10]Distance de Hamming (Mots binaire)*

La distance de Hamming $d(z, z')$, entre deux mots de même longueur z et z' (c.-à-d. le nombre de bit) est le nombre de symboles (positions) pour lesquels z et z' diffèrent.

Exemple 1.4.1 *Voici cette exemple*

$1^{ere} mot$	$2^{eme} mot$	*Distance de Hamming*
00000**1**1	00000**1**0**1**	1
0000**0111**	0000**1000**	4
0**1111111**	0**1**011**1**0	2
01010001	**10101110**	8

TAB 1.1 : *Calcule la distance de Hamming entre* 2 *mots binaire de* 8 *bits*

Les mots sont dits **adjacents** s'ils sont à une distance de Hamming de 1

Pour remédier à l'inconvénient de la distance de Hamming, on utilise un codage, appelé codage de Gray

Codage de Gray[10]

C'est un codage qui possède la propriété suivante :
Les éléments n et $n + 1$ seront codés de telle manière que leur distance soit égale à 1.

Exemple 1.4.2 *:*

Codage Décimale	*Codage Binaire naturel*	*Codage de Gray*
0	*0000*	*000**0***
1	*000**1***	*000**1***
2	*00**10***	*00**1**1*
3	*001**1***	*001**0***
4	*0**100***	*0**1**10*

TAB 1.2 : *distance de Hamming entre* 2 *mots binaire de* 8 *bits*

La fonction de décodage d

Elle doit permettre le passage de la représentation binaire vers la représentation en termes d'états de la variable initiale. Si cette variable prend des valeurs entières, nous devons avoir : $d : \{0,1\}^l \rightarrow \mathbb{N}$ (où l est la longueur de la chaîne). Le décodage le plus souvent retenu est un simple changement de base. par exemple la chaîne $A = \{0,0,0,1,1\}$ peut être décodée de manière à donner

$$0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3 \tag{1.4.2}$$

De manière plus générale, toute chaîne binaire A peut donc être décodée en une valeur entière x selon la règle suivante :

$$x = d(A) = \sum_{i=1}^{l} a_i 2^{l-i} \tag{1.4.3}$$

D'une façon générale il existe deux types de difficultés dans le choix d'un codage :

– La capacité de pouvoir s'adapter au problème de façon à limiter la taille de l'espace de recherche et engendrer les nouveaux chromosomes le plus possible (c'est à dire : Coder des solutions qui soient réalisables).

– Facilité de manipulation dans les opérations de génération de nouveaux gène.

1.4.2 Espace de recherche des solutions

La plupart des méthodes d'optimisation effectuent une recherche point à point. Les règles de transition d'un point à un autre sont souvent déterministes et la solution trouvée est souvent un optimum local au lieu d'être un optimum global. Les AGs, effectuent la recherche à partir d'une population de chaînes générées aléatoirement. Dans cette population, on retrouvera à la fois des candidats très performants et d'autres qui le sont moins. Le parallélisme induit

est un avantage évident car l'approche de la recherche à partir d'une population peut être perçue comme une recherche locale dans un sens généralisé. Ce n'est pas le voisinage d'une seule solution qui est explorée, mais le voisinage de toute la population.

1.4.3 Génération de la population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Une connaissance des solutions de bonne qualité comme point d'initialisation permet à l'algorithme de converger plus rapidement vers l'optimum ou du moins s'y rapprocher. Les individus sont alors générés dans un sous-domaine particulièrement proche de ces solutions de départ. Dans le cas où l'on ne dispose que peu d'informations sur le problème à résoudre, il est naturel de générer aléatoirement des individus, mais il est essentiel que la population initiale soit répartie sur tout le domaine de recherche. Tout en veillant à ce que les individus produit respectent les contraintes.
Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités. La diversité de la population doit être entretenue aux cours des générations afin d'explorer le plus largement possible l'espace de recherche. C'est le rôle des opérateurs de croisement et de mutation.

1.4.4 Taille de la population

La question la plus évidente concernant la mise en œuvre d'un AG est : quel est la taille de la population à utiliser ? Des populations de petites tailles entrainent le risque du la non couverture de l'espace des solutions en entier et, ceux de grandes tailles ralentissent, souvent, l'algorithme. Donc, la sélection de la taille du la population est importante, car elle la vitesse de convergence de l'algorithme. La diversité de la population est aussi importante. Pour prendre en considération ce qui vient d'être noté, nous devons respecter les deux règles suivantes :

1. Chaque gène doit être "représenté" dans la population de départ. Le passage d'une population à la suivante se fait par combinaisons des éléments de la première. Si cette dernière est incomplète cela entraine une recherche partielle dans l'espace des solutions réalisables. Très souvent un opérateur dit de mutation est appliqué pour palier à

l'incomplétude.

Un nombre minimum de membres pour représenter chaque gène dans la population initiale varie d'un problème à un autre. Pour l'exemple du problème du P-médian le nombre minimum choisi est donné[8] par :

$$PopSz = N/P \tag{1.4.4}$$

avec N :demande nœud et P : nombre de médians.

2. La taille de la population doit être proportionnelle au nombre de solutions. En général, plus la zone de faisabilité d'un problème est importante, plus il est difficile de trouver un meilleure choix.

Le but est de pouvoir choisir une taille de la population en mettant au point une formule qui tient compte des remarques suivantes :
Pour le problème du P-médian, la taille du problème $PopSz$ [8]

$$PopSz(n,p) = max\left\{2, \left[\frac{n}{100} \cdot \frac{\ln(S)}{d}\right]\right\} d \tag{1.4.5}$$

où $S = C_n^p$ le nombre de toutes les solutions réalisables du problème et $d = [n/p]$ la densité de la population. Chaque gène peut apparaitre avec une probabilité égale à au moins deux fois la densité. Le terme $Ln(S)$ mesure l'accroissement de la taille de la population en fonction de S . Ce taux d'accroissement est raisonnable ce qui permet de pouvoir gérer des populations de taille assez grande.
La taille de la population est en générale choisie comme constante durant tout le processus. Néanmoins on peut envisager de choisir une taille variable[11].

1.4.5 Fonction d'évolution (fitness function)

A chaque solution, on associe une fonction de performance (ou fitness function en terminologie anglo-saxonne) reliée à la valeur de la fonction objectif. Cette fonction performance décrit la performance de l'individu qui est représenté par un chromosome.
En effet, pour que les algorithmes génétiques se comportent bien, nous devons trouver une manière de formuler des fonctions performance ne comportant pas trop de maxima locaux.

Par conséquent la construction de la fonction performance est très importante, pour un problème de maximisation, la fonction performance peut être égale à la fonction Objectif. Par contre, pour les problèmes de minimisation, la fonction performance choisie est l'inverse de la fonction objectif. Dans tous les cas, l'algorithme génétique cherche à maximiser la fonction performance qui, dans le cadre d'un problème de minimisation, prend la forme suivante :

$$Fitness(x_i) = \frac{1}{f(x_i)} \tag{1.4.6}$$

Où $f(x_i)$ représente la fonction objective évaluée pour l'individu x_i.

- **Problème sous contraintes**

Les algorithmes génétiques, comme toutes autres méthodes classiques d'optimisation, trouvent des difficultés pour traiter les contraintes. Ces contraintes représentent, dans le cas d'un problème mécanique par exemple, les conditions technologiques qui doivent être satisfaites. Un problème sous contrainte est souvent formulé comme suit :

$$Maximiser \quad : \qquad Fitness(x) = \frac{1}{f(x)} \tag{1.4.7}$$

$$avec \quad : \qquad g_j(x) \geq 0 \qquad j = 1, \ldots, J \tag{1.4.8}$$

$$h_i(x) = 0 \qquad i = 1, \ldots, I \tag{1.4.9}$$

Où g représente des contraintes de type inégalité et h des contraintes de type égalité.
Plusieurs méthodes ont été proposées dans la littérature, les plus utilisées étant basées sur le principe de pénalité [9]. Avec ce principe, la formulation d'un problème sous contraintes peut s'exprimer, généralement, de la manière suivante :

$$Fitness(x) = \begin{cases} \frac{1}{f(x)} & si\ x \in \hat{E} \\ \frac{1}{f(x)+\mu P(x)} & si\ non \end{cases} \tag{1.4.10}$$

Où $\hat{E}$ est l'espace de solutions admissibles ; $\mu > 0$, en général un réel positif ,appelé paramètre de pénalisation et $P(x)$ la fonction de pénalisation exprimée en fonction des contraintes :

$$P(x) = \sum_{j=1}^{J} [\max\{0, g_j(x)\}]^2 + \sum_{i=1}^{I} |h_i(x)|^2 \quad (1.4.11)$$

Les paramètres de pénalisation dépendent généralement du problème traité et doivent être ajustés en fonction des paramètres du problème lui-même[9].

1.5 Les Opérateurs génétiques

Quatre mécanismes composent essentiellement les opérateurs génétiques : la sélection, le croisement, la mutation et l'insertion. Ces opérateurs se retrouvent dans la littérature sous plusieurs variantes.

1.5.1 Opérateur de Sélection

Les parents sont choisis aléatoirement suivant une méthode qui favorise les meilleurs d'entre eux (loi de sélection naturelle). C'est la phase dite de sélection basée sur le principe Darwinien de l'évolution. Certains individus peuvent être choisis plusieurs fois tandis que d'autres ne le seront jamais[5].

Cet opérateur est permet de définir quels seront les individus de P qui vont être dupliqués dans la nouvelle population $P\prime$ et vont servir de parents (application de l'opérateur de croisement). Soit N le nombre d'individus de P, on doit en sélectionner $N/2$ (l'opérateur de croisement nous permet de repasser à N individus). Plusieurs opérateurs de sélection existent parmi lesquels la sélection uniforme, la sélection par probabilité, par rang ou par tournoi ainsi que d'autres techniques faisant intervenir des notions de voisinage entre individus.

Sélection uniforme

Cette méthode, simple, consiste à sélectionner aléatoirement un individu, d'une manière uniforme sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité uniforme $(1/N)$ d'être sélectionné. La convergence de l'algorithme est en général lente.

Sélection par la roulette (loterie biaisée)

C'est la méthode la plus connue des sélections stochastiques, proposé par J. Holland [2]. Elle consiste à sélectionner les individus proportionnellement à leur performance, donc plus les individus sont adaptés au problème, plus ils ont de chances d'être sélectionnés. La probabilité de sélection est calculée à partir de la valeur de "fitness" du chromosome dans la population. Ainsi un individu x a la probabilité suivante d'être sélectionné :

$$P_{sel}(x_i) = \frac{Fitness(x_i)}{\sum_{i=1}^{PopSz} Fitness(x_i)} \tag{1.5.1}$$

Où $PopSz$ représente le nombre d'individus de la population.
(voir TAB 1.3)

	$Chromosome$	$Fitness$	$P_{sel}(x_i) = \frac{Fitness}{Total(Fitness)}$
	Individu 1	30	33 %
	Individu 2	60	60 %
	Individu 3	10	10 %
$Total$		100	100%

TAB1.3 : *Sélection par rang pour un problème de maximisation.*

Pour un problème de minimisation, la probabilité de sélection pour un individu i est donnée par :

$$P'_{sel}(x_i) = \frac{(1 - P_{sel}(x_i))}{(N-1)} \tag{1.5.2}$$

Les probabilités doivent être recalculées à chaque génération. Comme le montre *la Figure I.2*, La roue est divisée en autant de secteurs que d'individus dans la population. La taille de ces secteurs est proportionnelle à la fonction performance de chaque individu. En faisant tourner la roulette, l'individu pointé à l'arrêt de la boule est sélectionné. Les individus les mieux adaptés ont donc plus de chance d'être tirés au sort lors du déroulement du jeu.

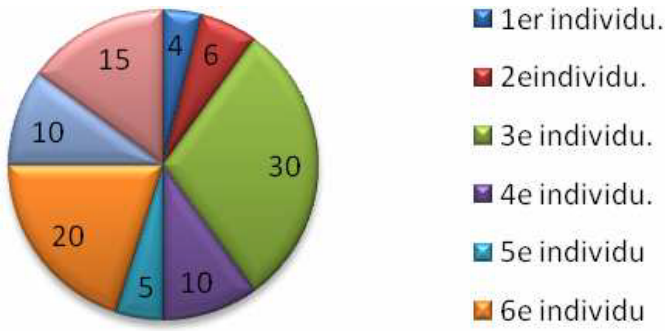


Figure 1.2 : *La roulette*

Cette méthode présente l'inconvénient majeur de domination d'un individu ("localement supérieur") par rapport aux autres. Ceci entraine une perte de diversité. On se retrouve face un problème des de "convergence prématurée" : On converge vers un optimum local.

Sélection par rang

Cette méthode consiste à ranger d'abord les individus selon un ordre (croissant ou décroissant de leurs performances) puis à attribuer (à l'aide d'une procédure) une probabilité de sélection en fonction de rang.
Cette probabilité est proportionnelle à la performance et est donnée la formule suivante :

$$P_{sel}(x_i) = \frac{Rang(x_i)}{\sum_{i=1}^{PopSz} Rang(x_i)} \tag{1.5.3}$$

Exemple :

	Chromosome	*Fitness*	*Rang*	$P_{sel}(x_i) = \frac{Rang}{Total(Rang)}$
	Individu 1	30	2	33.33%
	Individu 2	60	3	50%
	Individu 3	10	1	16.67%
Total		100	6	100%

TAB 1.4 *: Sélection par rang pour un problème de maximisation.*

On peut envisager d'autres variantes de classement des individus. Le choix des variantes dépend du problème traité.

Sélection élitiste

Cette méthode consiste à sélectionner les M meilleurs individus de la population P dont on a besoin pour la nouvelle génération P', après les avoir trié de manière décroissante selon leurs fonctions de performance. Il est inutile de préciser que cette méthode est encore pire que celle de la loterie biaisée dans le sens où elle amène à une convergence prématurée encore plus rapidement et surtout de manière encore plus sûre que la méthode de sélection de la loterie biaisée ; en effet, la pression de la sélection est trop forte, la variance nulle et la diversité inexistante, du moins le peu de diversité qu'il pourrait y avoir ne résultera pas de la sélection mais plutôt du croisement et des mutations.

Sélection par Tournoi

Cette méthode de sélection augmente les chances des individus de "mauvaise qualité" par rapport à leur fitness, de participer à l'amélioration de la population. En effet, c'est une compétition entre les individus d'une sous-population de taille M $(M \leq N)$ prise au hasard dans la population. Le paramètre M est fixé à priori par l'utilisateur. L'individu de meilleure qualité par rapport à la sous-population sera considéré comme vainqueur et sera sélectionné pour l'application de l'opérateur de croisement. Le paramètre M joue un rôle important dans la méthode du tournoi. Par conséquent le choix de M permet de faire varier la pression sélective. De cette manière, on contrôle les chances de sélection des individus les plus performants par rapport aux plus faibles.
Dans le cas où $M = N$ avec N est la taille de la population. Le résultat par la sélection de la méthode du tournoi donne à chaque fois un seul individu qui réduit l'algorithme génétique a un algorithme de recherche local travaillant sur une seule solution à la fois. Ce type d'algorithmes a pour inconvénient de converger parfois rapidement vers un optimum local.
Dans le cas $M = 1$, la méthode de sélection du tournoi correspond à la sélection aléatoire.

Des études comparatives entre les différentes méthodes de sélection ont montré que la sélection par rang et la sélection par tournoi donnent de meilleurs résultats. Ces méthodes ont été comparées sur des critères de qualité des résultats obtenus et de vitesse de convergence[**11**]**.**

1.5.2 Opérateur de croisement (crossover)

L'opérateur de croisement a pour but d'enrichir la diversité de la population en manipulant les composantes des chromosomes. Le principe de cette opérateur et de produire deux nouveaux individus, les enfants, en échangeant des informations entre deux autres individus appelés parents. Le croisement s'effectue en deux étapes. Dans la première étape, deux individus sont choisis aléatoirement dans la population sélectionnée. Dans la deuxième étape, en localisant d'une manière aléatoire des points sur les deux chromosomes parents, l'opération de croisement échange le contenu (informations) de ces points entre les deux chromosomes. Cette opération produit deux individus modifiés génétiquement qui font partie de la génération ultérieure. Ce sont les enfants. Le taux maximal de croisement est contrôlé par la probabilité de croisement Pc.

L'opérateur de croisement doit participer à la convergence de l'algorithme en produisant une population plus performante. Cet objectif ne peut être atteint en croisant aléatoirement des individus de la population. Le principe des algorithmes génétiques, inspiré de la biologie, consiste en effet à transmettre les caractéristiques des bons individus aux générations ultérieures. Afin de respecter ce principe, on doit favoriser le croisement des individus les plus performants. Cela peut être réalisé en gérant la probabilité de croiser deux parents Pr_1 et Pr_2. Nous proposons alors la formule suivante :

$$Pc(Pr_1,\ Pr_2) = Coef_c\ \frac{(PopSz - moy(r_1, r_2) + 1)}{PopSz} \tag{1.5.4}$$

Le paramètre $Coef_c$ est une constante entre [0.25, 0.85] , $PopSz$ est la taille de la population et $moy(r_1, r_2)$ la moyenne des rang r_1 et r_2 des parents Pr_1 et Pr_2 respectivement.

La formule (précédente) la probabilité de croiser deux parents est une fonction linéaire de la moyenne de leurs rangs. Si cette moyenne est faible, la probabilité de croisement sera forte. Cela signifie que deux parents performants, c'est-à-dire placés aux rangs les plus faibles, seront favorisés par l'opération de croisement. A l'inverse, deux parents peu performants classés dans une bonne position ont une faible probabilité de croisement. Dans une situation intermédiaire, le croisement, d'un parent performant avec un parent non performant, est associée une probabilité moyenne. Cela contribue à entretenir la diversité génétique sans en

écarter de manière définitive le croisement entre des individus performants et d'autres qui le sont moins.

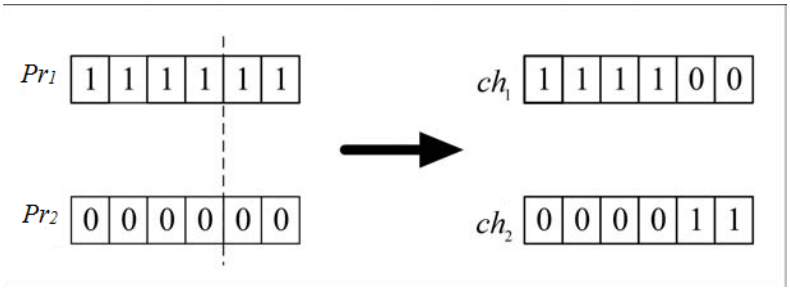
Notons que : $P_c \in]0, Coef_c[\ \forall (r_1, r_2)$. En effet, les rangs r_1 et r_2 prennent des valeurs entières comprises entre 1 et $PopSz$. Le rapport de l'équation est alors supérieur à 0 et inférieur à 1 et, par conséquent, une probabilité Pc comprise entre 0 et $Coef_c$. la valeur de $Coef_c$ a été choisie selon les tests numériques effectués.

Après on gèrera un nombre aléatoire $\alpha \in [0, 1]$, si $(\alpha \leq P_c)$, on applique l'opérateur de croisement sur le couple.

Pour conclure sur les opérateurs de croisement, nous en présentons quelques types :

Croisement à un ou à deux points

Son principe est de d'échanger les informations de deux chromosomes parents Pr_1 et Pr_2, la technique consiste, dans une première étape, à découper les deux chromosomes par un point, ensuite permuter le contenu des zones de chacun des chromosomes, ce qui produit deux enfants Ch_1 et Ch_2 comme indiqué sur la $Figure$ 1.3



$Figure$1.3 : *Croisement en un point*

Dans cette exemple, $Figure$ 1.3, un croisement localisé à la quatrième position a eu lieu entre les chromosomes Pr_1 et Pr_2 : il s'agit bien d'un croisement en un seul point.

Ainsi on peut étendre ce principe de combinaison en choisissant non pas un seul point, mais 2, 3, etc... Sur *la* $Figure$ 1.4 nous représentons un croisement en deux points,

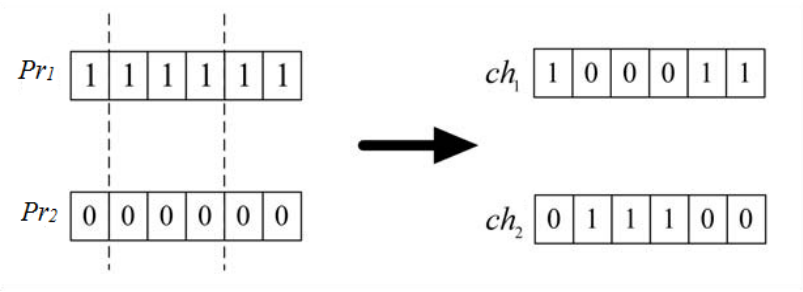


*Figure*1.4 : *Croisement en deux points*

Plusieurs études récentes ont montré que les croisements avec plusieurs points sont beaucoup plus avantageux que ceux à deux points. En effet, plus le nombre de points de croisements est grand et plus la probabilité de croisement est élevée.
Cela se traduit par plus d'échange de paramètres et donc plus d'échange d'informations.

Croisement uniforme

La mise en œuvre de ce procédé est fort simple, elle consiste à définir de manière aléatoire un "*Masque*", c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus, de quel parent le premier enfant Ch_1 devra hériter du gène s'y trouvant ; si l'un des locus de masque présente un $'0'$, l'enfant héritera le gène s'y trouvant du parent Pr_1, s'il présente un $'1'$ il en héritera du parent Pr_2. La création du deuxième enfant Ch_2 se fait de manière symétrique. Le schéma représentant le croisement uniforme est donné dans la (*Figure* 1.5)

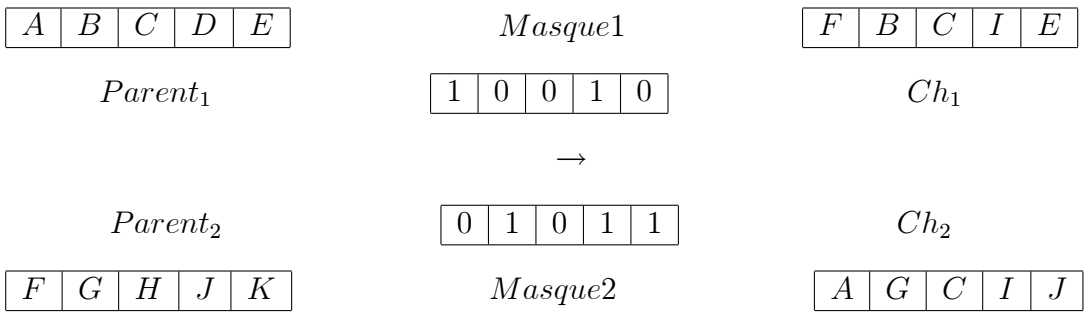


*Figure*1.5 : *Croisementuniforme*

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets et s'y prête moins pour les problèmes continus et les problèmes où l'on a recours à des codages non binaires.

Croisement barycentrique

Ce type de croisement est utilisé pour cas des problèmes continus. On sélectionne deux gènes Pr_1 et Pr_2 et on définit deux nouveaux gènes Ch et Ch_2 par combinaison linéaire :

$$\begin{cases} Ch_1 = a\ Pr_1 + (1-a)\ Pr_2 \\ Ch_2 = (1-a)\ Pr_1 + a\ Pr_2 \end{cases} \tag{1.5.5}$$

où a est un paramètre de pondération aléatoire qui prend généralement ses valeurs dans l'intervalle $[-0.5, 1.5]$.

Croisement entre plusieurs parents

Dans les problèmes de grande taille, l'exécution de l'opérateur de croisement entre deux parents seulement pourrait être peu efficace. Il est donc beaucoup plus efficace de procéder au croisement de plusieurs parents [7]. *La figure* 1.6 montre un croisement de deux points avec trois parents.

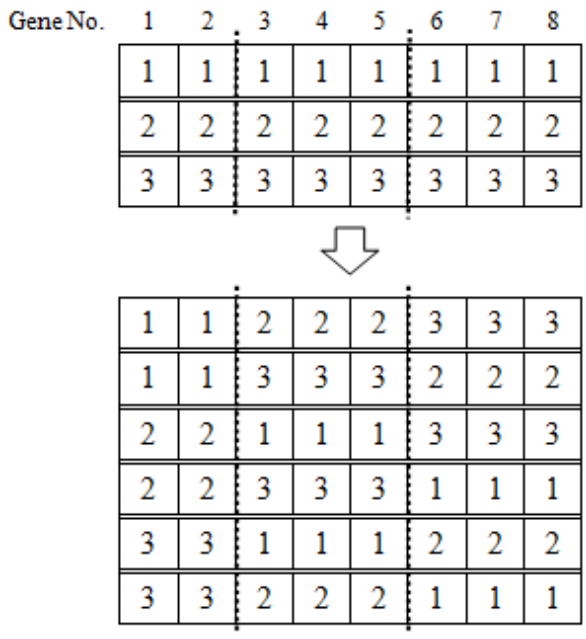


*Figure*1.6 : *Croisement entre plusieurs parents*

Il est difficile de trouver un type de croisement optimal. Il dépend du problème traite : Un opérateur de croisement universel (ou standard) n'existe pas et n'existera peut-être pas. La détermination d'un type d'opérateur de croisement est obtenue de façon expérimentale en observant l'effet de différents types de croisement.

1.5.3 L'opérateur de mutation :

Le but de la mutation est d'explorer, dans le domaine de conception, de nouvelles zones de solution afin de localiser l'optimum global. Le principe consiste à altérer une information contenue dans le tableau représentant le chromosome. L'exemple donné par la(*Figure* 1.7) représente une mutation du chromosome Ch en transformant son 5^{eme} gène de 1 en 0. Pour mettre en œuvre ce principe de manière pratique, plusieurs variantes sont possibles : locale, uniforme ou classique [11].

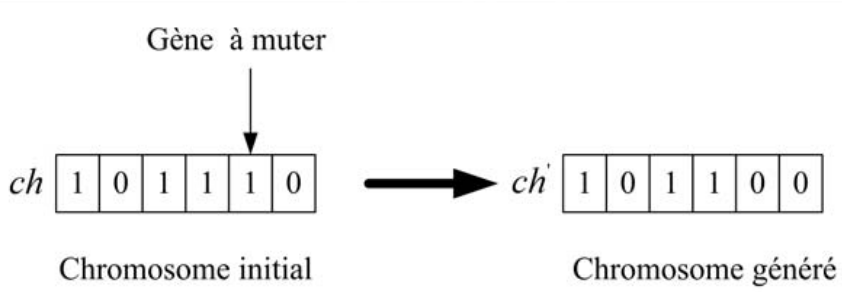


*Figure*1.7 : *Principe de l'opérateur de mutation*

Toutes ces variantes diffèrent les unes des autres mais ont en commun d'utiliser une probabilité de mutation qui peut être utilisée de manière aléatoire ou adaptative [4]. L'objectif de ce paragraphe est de justifier les choix de ses différentes variantes.

Les individus de la population peuvent être classés en trois catégories : les performants, les moins performants et les intermédiaires. En conséquence, ils ne doivent pas être exposés au même type de mutation.

Les plus performants représentent des optimaux locaux et une altération aléatoire des informations dont ils sont porteurs peut détériorer leurs caractéristiques. Pour cette catégorie

d'individus, nous proposons une mutation de type locale.

la mutation local : Ce type de mutation a besoin d'un opérateur local, conçu spécifiquement pour les problèmes d'optimisation de "forme" : Le principe consiste à faire des petites modifications localisées sur les contours du domaine de conception. Cependant, ce type de mutation ne permet pas d'explorer tout le domaine de conception.

La mutation uniforme : Elle est appliquée sur les individus les moins performants. Elle consiste à choisir le point de mutation aléatoire. Dans ce type de mutation, tous les points ont la même probabilité d'être muté. Mais l'aspect aléatoire de ce dernier type de mutation ne peut garantir une diversité génétique qui représente un facteur important dans un algorithme génétique.

La mutation classique : Ce troisième type de mutation est appliqué aux individus intermédiaires situés entre les plus et les moins performants. Elle est basée sur la diversité génétique. Chaque gène a une forte probabilité de mutation. En affectant à chaque point une probabilité de sélection, le principe de la roulette est appliqué pour choisir le point de mutation. Ce type de mutation améliore l'exploration et permet de maintenir la diversité génétique, mais il peut ralentir la convergence.

Dans ces deux types de mutation, uniforme et classique, la valeur au point de mutation est remplacée par une valeur choisie aléatoirement dans l'espace de recherche de la variable liée à ce point. L'objectif de ces deux types de mutation est de se rapprocher de l'optimum global en explorant le domaine de recherche. Mais elles permettent difficilement de converger précisément vers cet optimum lorsque des optima locaux coexistent dans son voisinage. Puisque les différences entre les optima locaux et l'optimum global sont naturellement localisés, les grandes tendances définissant la topologie étant identiques. C'est le rôle de la mutation locale qui permet de provoquer des altérations locales afin de pouvoir éviter des optima locaux pour converger vers l'optimum global. Ce dernier type de mutation joue aussi un rôle de lissage des frontières quand l'algorithme converge vers l'optimum global[4].

De manière pratique, l'opérateur de mutation consiste à altérer une information dans le tableau représentant le chromosome avec une probabilité P_m très faible, généralement comprise entre 0.01 et 0.001. Un individu x_i, choisi aléatoirement dans la population, mute si la

probabilité de mutation $P_m(x_i)$ vérifie la condition [11] :

$$rand \leq P_m(x_i) \tag{1.5.6}$$

Où $rand$ est une valeur générée aléatoirement dans l'intervalle $[0, 1]$.
L'exposition de tous les individus à une probabilité de mutation affecte aléatoirement le risque d'altérer les bonnes caractéristiques des individus performants. Or, ces bonnes caractéristiques doivent être transmises aux générations ultérieures par croisement. Pour éviter de les altérer par mutation, une approche de distribution linéaire de probabilité est utilisée pour favoriser la mutation des individus les moins performants :

$$P_m(x_i) = Coef_m.\left(\frac{rang(x_i)}{PopSz}\right) \tag{1.5.7}$$

Où $Coef_m$ est une constante ,appartenant à l'intervalle $[0, 1]$, déterminée de manière expérimentale. $rang(x_i)$ est le rang de l'individu x_i et $popSz$ la taille de population [7].
Cette approche, exprime la probabilité de mutation d'un individu en fonction de son rang. Cela consiste à une contribution originale par rapport aux distributions aléatoires de probabilités utilisées de manière classique. Dans la formule (*précédente*), les individus performants sont "protégés" alors que les individus les moins bons sont perturbés.

Cet opérateur dispose de 4 grands avantages :

- Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques.
- Il permet d'éviter un phénomène connu sous le nom de dérive génétique. On parle de dérive génétique quand certains gènes favorisés par le hasard se répandent au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. Le fait que l'opérateur de mutation puisse entraîner de manière aléatoire des changements au niveau de n'importe quel locus permet d'éviter l'installation de cette situation défavorable.
- Il permet de limiter les risques d'une convergence prématurée causés par exemple par une méthode de sélection élitiste imposant à la population une pression sélective trop forte. En effet, dans le cas d'une convergence prématurée on se retrouve avec une population dont tous les individus sont identiques mais ne sont que des optimums locaux. Tous les individus

étant identiques, le croisement ne changera rien à la situation. En effet, l'échange d'informations par croisement entre des individus strictement identiques est bien sûr totalement sans conséquence ; on aura beau choisir la méthode de croisement qu'on veut on se retrouvera toujours à échanger des portions de chromosomes identiques et la population n'évoluera pas. L'évolution se retrouvant bloquée on n'atteindra jamais l'optimum global.La mutation entrainant des inversions de bits de manière aléatoire permet de réintroduire des différences entre les individus et donc de nous extirper de cette situation.

– La mutation permet d'atteindre la propriété d' ergodicité (garantissant que chaque point de l'espace puisse être attient). Ce qui augmente les chances de pouvoir atteindre l'optimum global.

Les taux de croisement et de mutation sont contrôlés par probabilités respectives Pc et Pm. Ces probabilités sont définies par les relations $(1.5.4, 1.5.7)$ au cours de l'évolution de la population. Néanmoins, il est préférable de les ajuster de manière adaptative, en fonction de l'évolution de la population, plutôt que de les laisser fixes. C'est ce que nous allons aborder un peu plus loin.

1.5.4 Méthode d'insertion :

Après l'étape de mutation, on utilise une méthode d'insertion qui joue un rôle fondamental pour le passage d'une génération à une autre. Lors de la construction de cette population, on se trouve devant un vrai problème : faut-il garder les enfants au les parents ou bien un certain pourcentage des deux pour laisser la taille de la population (N) constante ?

Il s'agit de concevoir une stratégie d'évolution de la population. On distingue deux stratégies :

la première stratégie : notée ($\mathbf{N}$, $\mathbf{N_f}$), consiste à choisir les N individus à partir de N_f enfants déjà crées par les opérateurs de croisement et de mutation. Dans cette stratégie, on suppose que $N_f \geq N$. quand $N_f = N$, nous parlerons de la méthode générationnelle qui remplace les parents par les enfants ;

la seconde stratégie : notée ($\mathbf{N}+$ $\mathbf{N_f}$), consiste à choisir les N individus à partir de N parents de la population précédente et de N_f nouveaux enfants. Un cas particulier de cette stratégie, appelé l'insertion élitiste qui consiste à sauvegarder une grande partie de la population dans la génération suivante. A chaque itération quelques chromosomes (parents) ayant

les meilleurs coûts seront sélectionnés afin de créer des chromosomes fils qui remplaceront les plus mauvais parents. Le reste de la population survie et sera copié dans la nouvelle génération.
L'objectif de cette insertion est d'éviter la déperdition des meilleurs chromosomes soient perdus après les opérations de croisement et de mutation. Cette méthode améliore considérablement les algorithmes génétiques, car elle permet de conserver, à une itération K, les meilleurs individus trouvés dans toutes les populations générées antérieurement.

1.6 Tests D'arrêt

Le critère d'arrêt indique que la solution est suffisamment proche de l'optimum. Plusieurs critères d'arrêt de l'algorithme sont possibles.
On peut arrêter l'algorithme après un nombre suffisant de générations pour que l'espace de recherche soit convenablement exploré [6]. Ce critère peut s'avérer coûteux en temps de calcul si le nombre d'individus à traiter dans chaque population est important.
On arrête l'algorithme lorsque l'on estime que la population n'évolue pas assez rapidement. On peut supposer qu'on est suffisamment proche de l'optimum.

Il est à noter qu'aucune certitude concernant la bonne convergence de l'algorithme n'est assurée. Comme dans toute procédure d'optimisation l'arrêt est arbitraire, il nous faut donc exécuter plusieurs fois l'algorithme et de procéder à une analyse statistique des résultats. La solution obtenue ''en un temps fini" ne constitue qu'une approximation de l'optimum.

1.7 Les avantages et les inconvénients

Par rapport aux algorithmes classiques d'optimisation, l'algorithme génétiques présente plusieurs points forts comme :

– Le fait d'utiliser seulement l'évaluation de la fonction objectif sans se soucier de sa nature. En effet nous n'avons besoin d'aucune propriété particulière sur la fonction à optimiser

(continuité, dérivabilité, convexité, etc.), ce qui lui donne plus de souplesse et un large domaine d'applications ;

– Génération d'une forme de parallélisme en travaillant sur plusieurs points en même temps (population de taille N) au lieu d'un seul itéré dans les algorithmes classiques, donc il est efficace lorsque cette fonction possède plusieurs optimums locaux ;

– L'utilisation des règles de transition probabilistes (probabilités de mutation, ...), contrairement aux algorithmes détermines où la transition entre deux itérations successives et imposée par la structure et la nature de l'algorithme. Cette utilisation permet, dans certaines situations, aux algorithmes génétiques d'éviter des optimums locaux et de se diriger vers un optimum global ;

Malgré leurs efficacités dans la résolution des problèmes d'optimisation, les algorithmes génétiques n'ont pas encore totalement démontré leurs preuves mathématiques. Cela est essentiellement dû à la difficulté d'adopter une représentation conforme au problème traité et le grand nombre de paramètres qui doivent être fixés par l'utilisateur (taille de la population, probabilités d'application des opérateurs génétiques, ...) et dont la mise au point de ces paramètres ne peut se faire qu'empiriquement. Pour effectuer cette mise au point, il est parfois nécessaire de procéder à de nombreux tests numériques.

1.8 Conclusion

Dans cette partie, nous avons présenté les algorithmes génétiques de manière générale et donné les éléments de base nécessaires à la compréhension de la méthode.
La partie suivante sera consacrée à l'application de l'algorithme génétique au problème classique de l'optimisation combinatoire qu'est le P-médian dans un graphe. Mais avant d'entamer cette dernière, nous donnons un peu aperçu de l'analyse théorique.

Chapitre 2

L'analyse théorique des algorithmes génétiques

2.1 Introduction

Les algorithmes génétiques ont monté leur efficacité pratique bien avant que les résultats de convergence théorique ne soient établis. Nous disposons maintenant d'une approche théorique qui permet de mieux comprendre le fonctionnement des algorithmes génétiques,
Nous présenterons, tout d'abord, quelque principe classique pour améliorer le processus de sélection. Puis, nous nous intéresserons à la théorie des schémas qui développée par J.Holland, elle étudie le comportement asymptotique de l'algorithme binaire et l'e et des différents opérateurs sur la structure des schémas, que nous ne détaillerons pas la suite.

2.2 Améliorations classiques

Les processus de sélection présentés sont très sensibles aux écarts d'adaptation entre les individus. En effet, dans certains cas, un très bon individu risque d'être reproduit trop souvent et peut même provoquer l'élimination complète de ses congénères ; on obtient alors une population contenant un seul type d'individu (convergence prématurée).
Pour éviter ce comportement, il existe d'autres modes de sélection ainsi que des principes (scaling, sharing) qui empêchent les individus forts d'éliminer complètement les plus faibles.

2.2.1 Scaling

Le scaling ou mise à l'échelle, modifie les fitness afin de réduire ou d'amplifier artificiellement les écarts entre les individus. Le processus de sélection n'opère plus sur la fitness réelle mais sur son image après scaling. On souhaite ainsi aplatir ou dilater la fonction d'évaluation. La perte de précision qui s'ensuit est au prix d'une meilleure convergence.
Parmi les fonctions de scaling, on peut envisager le scaling linéaire et le scaling exponentiel.

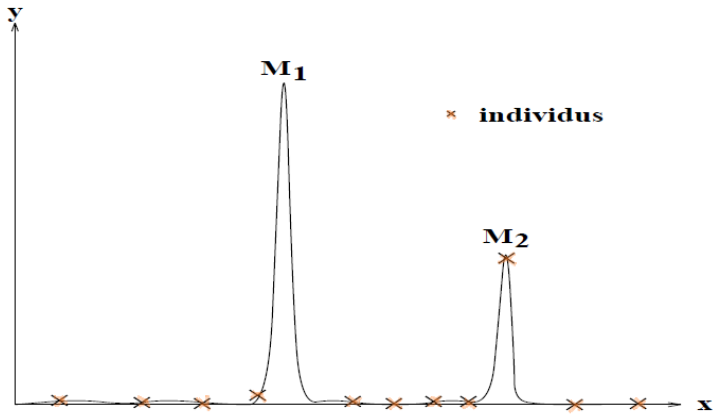


Figure 2.1 : *Exemple où les sélections classiques risquent de ne reproduire qu'un individu.*

Scaling linéaire

Dans ce cas la fitness initiale f_r est redéfinie en une nouvelle fitness f_s par l'opération suivante :

$$f_s = a\ f_r + b \tag{2.2.1}$$

En règle générale, le coefficient a est inférieur à un, ce qui permet de réduire les écarts de fitness et donc de favoriser l'exploration de l'espace.

Scaling exponentiel

Il est défini de la façon suivante :

$$f_s = (f_r)^{n(k)} \tag{2.2.2}$$

généralement $n(k)$ varier entre des faibles valeurs vers de fortes valeurs au cours des générations. Pour cela on utilise la formule :

$$n(k) = \alpha_1 \left(\tan \left[\frac{k}{K+1} \right] \frac{\pi}{2} \right)^{\alpha_2} \tag{2.2.3}$$

k étant la génération courante, K le nombre total de générations prévues, et α_1 et α_2 sont des paramètres à choisir. L'évolution de $n(k)$ en fonction de la génération k est donnée par *la figure* 2.2.

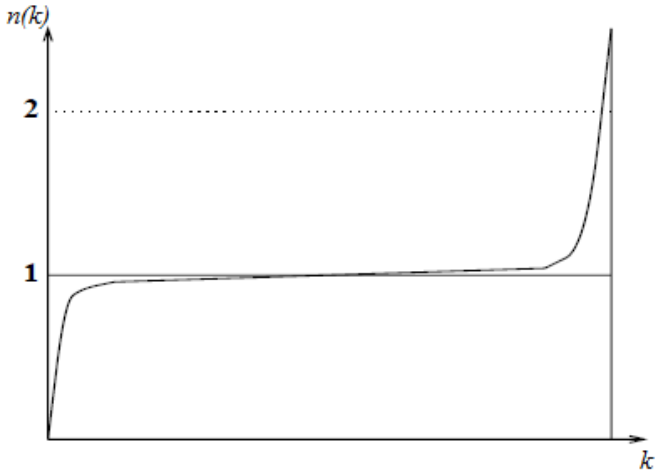


figure 2.2 :*l'évolution de* $n(k)$ *en fonction des générations*

Pour n(k) proche de zéro, on réduit fortement les écarts de fitness ; aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un algorithme de recherche aléatoire et permet d'explorer l'espace.

Pour n(k) proche de 1 ; le scaling est inopérant.

Pour n(k) $>$ 1 les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

Ce dernier principe de scaling donne dans la pratique des meilleurs résultats que le scaling linéaire. Donc le scaling permet une bonne exploration de l'espace d'état, mais ne favorise pas la répartition des individus sur les différents optima de la fonction objective. Cette répartition souhaitable est obtenue en utilisant l'opérateur de partage.

2.2.2 Partage (sharing)

L'objectif du partage est de répartir sur chaque sommet de la fonction à optimiser, un nombre d'individus proportionnel à la fitness associée à ce sommet. *La figure* 2.3 présente deux exemples de répartitions de populations dans le cas d'une fonction à cinq sommets : le premier avec partage, le second sans partage.

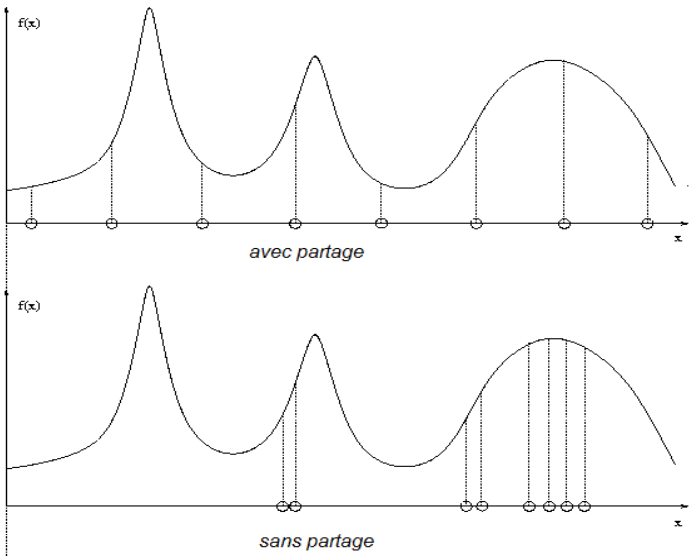


figure 2.3 : *Objectif du partage*

De la même façon que le scaling, le partage consiste à modifier le fitness utilisé par le processus de sélection. Pour éviter le rassemblement des individus autour d'un sommet dominant, le partage pénalise les fitness en fonction du taux de concentration de la population dans le voisinage d'un individu. Plus cet individu est entouré, plus on pénalisera sa reproduction. Alors on calcule notre nouvelle fitness de la façon suivante :

$$f'(x_i) = \frac{f(x_i)}{m'_i} \qquad où \qquad m'_i = \sum_{j=1}^{N} S(d(x_i, x_j)) \tag{2.2.4}$$

avec

$$S(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} & si \quad d < \sigma_{share} \\ 0 & si \quad d > \sigma_{share} \end{cases} \tag{2.2.5}$$

Ce calcul du terme pénalisant (au dénominateur) m'_i est très proche d'un calcul de densité par estimation non paramétrique. Il joue le rôle d'estimateur de la densité autour de l'individu

(x_i). la fonction S faisant office de noyau et σ_{share} de fenêtre. Il s'agit donc d'un opérateur qui pondère la fitness d'un individu par l'inverse de sa densité. On pénalise donc bien les individus se situant dans une zone "dense". Le paramètre σ_{share} permet de délimiter le voisinage d'un point et dépend du problème traité.
Dans la pratique ce type de partage donne effectivement de bons résultats mais au prix de N^2 calculs de distances entre chromosomes à chaque génération pour une population de taille N.

2.3 Théorie des schémas

Historiquement, les algorithmes génétiques binaires sont les plus anciens et ils ont donc été les plus étudiés sur le plan théorique. Hollond est le premier à proposer une version de la théorie des schémas, puis elle a été largement développée par Goldberg. Nous donnons dans ce qui suit quelques définitions fondamentales :

Définition 2.3.1 ***(Séquence)*** *On appelle séquence A de longueur $l(A)$ une suite $A = a_1a_2\dots\ a_l$, $\forall\ i \in\ \{1,...,l\ \}$, $a_i\ \in\ V = \{0,1\}$. Ceci correspond à la notion de séquence de bits que nous avons utilisée jusqu'à présent.*

Définition 2.3.2 ***(Schéma)*** *On appelle schéma H de longueur l une suite $H = a_1a_2\dots\ a_l$, $\forall\ i \in\ [1,l\ ]$, $a_i\ \in\ V+ = \{0,1,*\}$. Comme nous allons le voir, le signe $(*)$ en position i signifie que a_i peut être indifféremment un 0 ou un 1.*

Définition 2.3.3 ***(Instance)*** *On dit qu'une séquence $A = a_1a_2\dots\ a_l$ est une instance d'un schéma $H = b_1b_2\dots\ b_l$ si pour tout i tel que $b_i \neq *$ on a $a_i = b_i$.*

Ainsi, $H = 010*0101$ est un schéma et les séquences 010**0**0101 et 010**1**0101 sont des instances de H.

Définition 2.3.4 ***(Position fixe, position libre)*** *Soit un schéma H. On dit que i est une position fixe de H si $a_i = 1$ ou $a_i = 0$. On dira que i est une position libre de H si $a_i = *$.*

Définition 2.3.5 ***(Ordre d'un schéma)*** *On appelle ordre du schéma H , noté $o(H)$, le nombre de positions fixes de H.*

Par exemple, le schéma $H = 01 * *10 * 1$ a pour ordre $o(H) = 5$, le schéma $H\prime = * * * * *101$ a pour ordre $o(H\prime) = 3$. On remarquera qu'un schéma H de longueur $l(H)$ et d'ordre $o(H)$ admet $2^{(l(H)-o(H))}$ instances différentes.

Définition 2.3.6 ***(Longueur fondamentale)*** *On appelle longueur fondamentale du schéma H la distance séparant la première position fixe de H de la dernière position fixe de H. On note cette longueur fondamentale $l(H)$.*

Ainsi, le schéma $H = 1 * *01 * **$a pour longueur fondamentale $l(H) = 5 - 1 = 4$, le schéma $H' = 1 * * * * * *1$ a pour longueur fondamentale $l(H') = 8 - 1 = 7$, et pour le schéma $H'' = * * 1 * * * **$ nous avons $l(H'') = 3 - 3 = 0$.

Définition 2.3.7 ***(Adaptation d'une séquence)*** *On appelle adaptation d'une séquence A une valeur positive que nous noterons $f(A)$. f est la fonction objectif ou fitness du problème à résoudre.*

Définition 2.3.8 ***(Adaptation d'un schéma)*** *On appelle adaptation d'un schéma H la valeur*

$$f(H) = \frac{\sum_{i=1}^{2^{(l(H)-o(H))}} f(A_i)}{2^{(l(H)-o(H))}} \tag{2.3.1}$$

où les A_i décrivent l'ensemble des instances de H. L'adaptation d'un schéma est donc la moyenne des adaptations de ses instances.

2.3.1 Effets de la reproduction

Soit un ensemble $S = \{A_1, \ldots, A_i, \ldots, A_n\}$ de n séquences de bits tirées aléatoirement. Durant la reproduction, chaque séquence A_i se reproduira avec une probabilité :

$$p_i = \frac{f(A_i)}{\sum_{i=1}^{n} f(A_i)} \tag{2.3.2}$$

Supposons que nous ayons à l'instant t un nombre $m(H,t)$ d'instances représentants le schéma H dans la population S. À l'instant $t+1$ nous devons statistiquement avoir un nombre :

$$m(H,t+1) = m(H,t).n.\frac{f(H)}{\sum_{i=1}^{n} f(A_i)} \tag{2.3.3}$$

Posons

$$\overline{f}_t = \frac{\sum_{i=1}^{n} f(A_i)}{n} \tag{2.3.4}$$

$\overline{f}_t$ représente la moyenne de l'adaptation des séquences à l'instant t. La formule précédente devient :

$$m(H,t+1) = m(H,t)\frac{f(H)}{\overline{f}_t} \tag{2.3.5}$$

Posons

$$c_t(H) = \frac{f(H)}{f_t} - 1 \tag{2.3.6}$$

On obtient alors :

$$m(H,t+1) = (1 + c_t(H))\ m(H,t) \tag{2.3.7}$$

On voit donc qu'un schéma, dont l'adaptation est au-dessus de la moyenne, voit son nombre de représentants augmenter, suivant une progression qui est de type géométrique si nous faisons l'approximation que $c_t(H)$ est constant dans le temps. Nous avons alors :

$$m(H,t) = (1 + c(H))^t\ m(H,0) \tag{2.3.8}$$

Le résultat est donc clair : si la reproduction seule était en jeu, les schémas forts élimineraient très rapidement les schémas faibles.

2.3.2 Effet de croisement

Nous allons nous intéresser à la probabilité de survie $p_s(H)$ d'un schéma H lors d'une opération de croisement. Considérons cela sur un exemple. Soit le schéma $H = **10*1**$. Supposons qu'une séquence qui est une instance de ce schéma soit croisée avec une autre séquence. Quelle est la probabilité pour que la séquence résultante soit encore une instance de H ? Il est impossible de répondre exactement à la question, tout au plus peut-on donner une borne inférieure de cette valeur. Il est clair que H ne sera pas détruit si le site de croisement qui est tiré au sort est inférieur à 3 (avant le premier 1) ou s'il est supérieur à 5 (après le dernier 1).
On voit donc immédiatement qu'une borne inférieure de la probabilité de détruire un schéma H est $\frac{l(H)}{l-1}$. Donc la probabilité de survie dans un croisement est $1 - \frac{l(H)}{l-1}$. Si d'autre part on ne croise qu'une fraction p_c de séquences dans une population donnée, la probabilité de survie est donnée par :

$$p_s \simeq 1 - p_c \frac{l(H)}{l-1} \tag{2.3.9}$$

En rassemblant ce résultat et celui du paragraphe précédent, nous obtenons pour l'évolution d'une population :

$$m(H, t+1) \simeq m(H, t)\ (1 + ct(H))\ \left(1 - p_c \frac{l(H)}{l-1}\right) \tag{2.3.10}$$

2.3.3 Effet de mutation

Supposons que la probabilité de mutation d'un bit dans une séquence soit p_m. Dans un schéma H, seules les positions fixes peuvent être détruites. Comme la probabilité de survie

d'un bit est $1 - p_m$, la probabilité de survie d'un schéma H contenant $o(H)$ positions fixes est $(1 - p_m)^{o(H)}$. La probabilité de mutation étant toujours petite devant 1, on peut faire un développement limité au premier ordre, ce qui nous donne une probabilité de survie de $1 - o(H)\ p_m$.

En introduisant ce terme, nous avons donc l'équation finale :

$$m(H, t+1) \simeq m(H, t)\ (1 + ct(H))\ \left(1 - p_c \frac{l(H)}{l-1} - o(H) P_m\right)$$

On conclut de ci-dessus deux choses :

– les schémas qui ont une longueur fondamentale petite sont plus favorisés que les autres, lors de la génération d'une nouvelle population ;

– les schémas qui ont un ordre petit sont plus favorisés que les autres, lors de la génération d'une nouvelle population.

Ceci enseigne une chose fondamentale pour le codage de données : les schémas qui codent les données " intéressantes" pour le problème doivent avoir un ordre et une longueur fondamentale faibles, alors que les données " sans intérêt " doivent être codées par des schémas qui ont un ordre et une longueur fondamentale élevés[9].

2.4 Conclusion

Dans cette section, nous présentons quelques éléments de la théorie des algorithmes génétiques pour en déduire des résultats substantiels. Les résultats présentés ci-dessus ne sont qu'une introduction à la théorie des schémas. Il existe de nombreux approfondissements sur les principaux modèles théoriques de la théorie des schémas et ses extensions. La question qui reste encore posée est : Comment trouver de nouveaux algorithmes génétiques encore plus efficaces ? Autant de questions pour lesquelles nous trouverons sans doute des réponses dans de travaux futurs.

Chapitre 3

Exemple mettant en œuvre tous les opérateurs

L'exemple suivant va permettre de mettre en œuvre sur une génération tous les opérateurs étudies de manière à pouvoir observer le résultat de l'application successive de ces opérateurs. Cet exemple est dû à Goldberg (1989). Il consiste à trouver le maximum de la fonction $f(x) = x$ sur l'intervalle $[0, 31]$ où x est un entier naturel. On a 32 valeurs possibles pour x on choisit donc un codage discret sur 5 bits : on obtient donc la séquence $0, 1, 1, 0, 1$ pour 13, la séquence $1, 1, 0, 0, 1$ pour 25, etc...

On initialise la population initiale de manière aléatoire et on fixe sa taille à 4 individus. On définit simplement la fitness comme étant la valeur de x, vu qu'on en cherche la valeur maximum sur l'intervalle $[0, 31]$ plus la valeur de x sera élevée plus on se rapprochera du maximum de la fonction identité et donc plus la fitness sera grande. Soit la population initiale suivante :

Individu	Séquence	Fitness	% du total
1	01011	11	18.6
2	10011	19	32.2
3	00101	5	8.5
4	11000	24	40.7
Total		59	100

*figure*3.1 : *la population initiale*

On opte pour une sélection par la méthode la loterie biaisée :

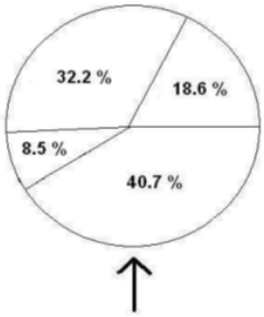


figure 3.2 : *Application de la mthode de slection de la loterie biaise sur la population*

On fait tourner la roue 4 fois de suite, en général on fait tourner $n/2$ fois, soit 2 fois dans ce cas, mais le nombre 2 étant trop petit on décide de la faire tourner 4 fois. On obtient la nouvelle population :

Individu	Séquence
1	11000
2	00101
3	11000
4	01011

figure 3.3 *: Individus sélectionnés par la méthode de la loterie biaisée*

On applique l'opérateur de croisement en utilisant un seul point de croisement. Normalement chaque couple donne 2 enfants qu'on ajoute à notre nouvelle population $P\prime$ la faisant passer de $n/2$ individus à n individu, mais vu que dans le cas de notre exemple nous avons déjà atteint nos n individus, les 2 enfants du couple remplaceront leurs parents. Deux couples sont formés de manière aléatoire :

• couple 1 : l'individu 2 avec l'individu 3

• couple 2 : l'individu 1 avec l'individu 4.

Les points de croisement sont eux aussi tiré au hasard.

On obtient le résultat suivant :

Parents	Enfants
00101	01011
01011	00101
11000	01011
01011	11000

figure 3.4 *: Résultat de l'application de l'opérateur de croisement avec un point de croisement sur les individus sélectionnés par la loterie biaisée*

On applique l'opérateur de mutation qui choisit de manière aléatoire si on doit faire une mutation et sur quel locus la faire :

Chromosome avant mutation	Chromosome après mutation
01011	11011
00101	00101
01011	01111
11000	11000

figure 3.5 *: Résultat de l'application de l'opérateur de mutation sur les individus engendrés par croisement*

Puis on applique l'opérateur de remplacement qui décide de remplacer 100% de la population P, la population P est donc entièrement remplacée par $P\prime$ et sa taille reste fixe :

Individu	Séquence	Fitness	% du total
1	11011	27	38
2	00101	5	7
3	01111	15	21.1
4	11000	24	33.8
Total		71	100

figure 3.6 : *la nouvelle population après application des différents opérateurs*

En une seule génération le maximum est passé de 24 à 27, et la fitness globale de la population a relativement augmentée pour passer de 59 à 71. On s'arrête ici pour cet exemple mais dans la réalité on continuera à engendrer des générations successives jusqu'à obtenir le maximum global : 31.

Le seul intérêt de cet exemple était de montrer une trace simple, sur une génération, de l'application successive des différents opérateurs. Il est bien sûr évident qu'un tel "problème" ne nécessitait pas l'utilisation d'algorithmes génétiques.

Chapitre 4

Application de l'algorithme génétique à un problème du P-médian

4.1 Introduction

La recherche opérationnelle en matière de localisations d'activités a été l'une des préoccupations majeures dans beaucoup de champs d'applications (logistique, transport, grande distribution, services bancaires, assurance). En particulier, en ce qui concerne la localisation de points de vente ou de services ou même d'entrepôts, une problématique importante est de trouver la localisation de P d'activités devant fournir n clients de telle manière que la somme de l'ensemble des distances séparant chaque activité aux clients les plus proches soit minimale. Ce problème est connu sous le nom du problème du P-médian.

Début de $XXème$ siècle dans les réflexions d'Alfred Weber sur la meilleure manière de placer un centre de production par rapport aux sources de matières premières. En pratique, le problème du P-médian est soulevé dans la plupart des réseaux qu'ils soient routiers, aériens ou téléphoniques. Handler et Mirchandi , Hakimi et Maheshwari ont dressé la liste très variée des applications potentielles du modèle P-médian comme les décisions de localisation pour les services d'urgence (police, pompiers, urgences médicales), les réseaux de communication et informatique (localisation des fichiers informatiques sur une série de serveurs identifiés), les applications militaires (centres militaires stratégiques), les activités de service public ou privé (les magasins, centre commerciaux, postes), l'intelligence artificielle. Ces plusieurs applications, c'est l'une des raisons pour laquelle notre travail s'est concentré sur le problème du P-médian.

4.2 Présentation du problème du P-médian

On suppose qu'on ait à satisfaire N demandes localisées pour un certain service. Ce service est effectué par des points qu'on appelle "Centre de services". Le problème consiste à déterminer la localisation optimale de P centres de services (P variant de 1 à N) de telle sorte que le "coût de satisfaction" de l'ensemble des demandes soit le plus petit possible.
Sachant que chaque demande est servie par le centre le plus "proche" et que chaque centre a une capacité infinie*[1]*.

4.3 Formulation Mathématique

Soit un graphe $G = (V, E)$ non orienté qu'on supposera, sans perte la généralité, simple et sans boucle. Soit $N = |v|$ l'ordre du graphe. A tout sommet v de$V$ on associe un nombre non négatif $w(v)$ (appelé poids de v) et à toute arête e de E est associée un nombre non négatif $C(e)$ (appelé coût de e).
Soit $X = \{x_1, x_2, ..., x_p\}$ un ensemble de P points du graphe. Nous entendons par un point de G, tout sommet x de graphe où, tout point x située sur une arête e de E (nous faisons l'hypothèse que le coût est proportionnel à la distance). La distance entre un sommet v et X est :

$$d(v, X) = Min\{d(v, x)\} \tag{4.3.1}$$

où d(v,x) représente la distance d'un plus court chemin entre v et x. on définit :

$$Q(X) = \sum_{v_{\in V}} w(v).d(v, X) \tag{4.3.2}$$

Un ensemble X^* minimisant Q est dit p-médian

Remarque 4.3.1

Si $P = N$ le problème du p-médian est évident $X^* = V$.
Si $P > N$ le problème du n'a pas de signification mathématique.

On supposera dans tout ce qui suit que P vérifie : $1 \leq P \leq N$

4.4 Formulation du problème en un programme linéaire

Soient :

d_{ij} : le coût unitaire de "satisfaction" du sommet j par i

w_j : la demande du sommet j

En générale, on pose $c_{ij} = w_j.d_{ij}$: coût de "satisfaction" de la demande totale du sommet j par le médian i

On définit les variables par :

$$x_{ij} = \begin{cases} 1 \text{ si } j \text{ est servi par le sommet } i & i,j \in V \\ 0 \end{cases} \quad (4.4.1)$$

$$x_{ij} = \begin{cases} 1 \quad \text{si } i \text{ est choisi comme médian} & i,j \in V \\ 0 \end{cases}$$

Le problème du p-médian peut être formulé en un programme linéaire (PL) suivant :

$$Min(z) = \sum_{i \in V} \sum_{j \in V} w_j.d_{ij}.x_{ij} \quad (4.4.2)$$

$$\sum_{i \in V} x_{ij} = 1 \quad ,\forall j \in V \quad (4.4.3)$$

$$\sum_{i \in V} x_{ii} = P \ ,\forall i \in V \quad (4.4.4)$$

$$x_{ij} - x_{ii} \leq 0, \quad \forall i,j \in V \quad (4.4.5)$$

$$x_{ij} \in \{0,1\}, \quad \forall i,j \in V \quad (4.4.6)$$

Interprétation des contraintes :

4.4.2 : Fonction objectif du problème du p-médian

4.4.3 : Signifie qu'un sommet j n'est servi que par un et un seul médian

4.4.4 : Signifie que le nombre de médians est fixé à P

4.4.5 : Signifie qu'un sommet j n'est servi par un sommet i que si i est choisi comme médian

Le problème du P-médian est connu pour être un problème NP-complets[2]. En effet, selon une logique d'analyse combinatoire, le nombre de solutions à examiner est fonction de n, le nombre de nœuds, et P, le nombre de médians à ouvrir :

$$\frac{n!}{p!(n-p)!} \tag{4.4.7}$$

On distingue deux types de méthodes de résolution des problèmes du P-médian (problèmes d'optimisation combinatoire) :

- Les méthodes exactes baseés sur le principe des méthodes de sélection et évaluation progressive (Branch and bound). Ces méthodes sont difficilement applicables pour des problèmes de grandes tailles.
- Les méthodes approchées qui nous fournissent des solutions approximatives. Ces méthodes se divisent à leurs tours en :

  1. Procédures spécifiques (Maranzana, Teitz & Bart) méthodes qui exploitent les propriétés que recèlent les problémes du P-médian. Ces méthodes sont très efficaces pour des problèmes de taille raisonnable.
  2. Procédures méthaheuristiques : Ces méthodes générales, s'adaptent parfaitement aux problèmes de grandes tailles. Parmi ces procédures nous citons l'algorithme génétique, objet de notre étude.

4.5 La résolution du modèle du p-médian en utilisant AG

4.5.1 Codage

Nous utilisons un codage simple où les gènes d'un chromosome correspondent aux indices des facilités sélectionnés. Par exemple, (5, 7, 2, 12) est un chromosome qui correspond à une solution possible pour un problème du 4-médian où les points de demande 2, 5, 7 et 12 sont choisies comme des médians.

4.5.2 La population initiale

Pour générer la population initiale, nous choisissons au hasard p nœuds puis on procède à l'affectation de chaque demande vers le médian le plus proche. La taille de la population est déterminée par la formule 1.4.5

4.5.3 La fonction d'évaluation et la sélection

La fonction d'évaluation est égale à la fonction objectif du problème du P-médian. Alors que les parents que nous avons sélectionnés sont choisis par des mécanismes de sélection biaisée.

4.5.4 Générer de nouveaux membres

La génération d'une nouvelle population s'obtient en appliquant les opérateurs de croisement, de mutation et éventuellement une opération d'insertion.

Opérateur de croisement

Contrairement à l'opérateur du croisement classique qui opère sur deux individus pour obtenir deux enfants, pour le problème du P-médian, l'opérateur du croisement agit, de la même façon que précédemment, sur deux parents mais pour ne générer qu'un seul fils. Le procédé de croisement est décrit comme suit :

1. Les gènes qui se trouvent chez les deux parents sont automatiquement sélectionnés pour former un nouveau chromosome "incomplet" (la solution obtenue est en général non réalisable).
2. Pour compléter on choisit aléatoirement des gènes dans l'un au l'autre des parents pour obtenir un chromosome complet à p gènes (on rend la solution réalisable).

Opérateur de mutation

Cet opérateur, utilisé pour assurer la diversité, procède de la manière suivante :

1. D'abord on choisit un nombre de candidats (nombre qui dépend du problème traité).
2. Pour tout candidat, choisir aléatoirement une position k entre 1 et p et changer le k^{eme}gène par un gène qui ne se trouve pas dans le chromosome.

Insertion

Cet opérateur est généralement exécuté pour obtenir une population de la taille voulue et assurer une diversité.

4.5.5 Critère d'arrêt

plusieurs critères d'arrêt peuvent être envisagés tels que :

– Exécuter l'algorithme pendant un nombre d'itération maximum $MaxIter$.

– On peut s'arrêter au bout d'un certain temps CPU.

– S'arrêter, si au bout d'un certain nombre d'itérations AC (égale à $[n\sqrt{p}]$ pour le problème du P-médian) la fonction objectif ne s'améliore pas[8].

4.6 L'Algorithme

L'algorithme en général peut être décrit comme suit :

Etape 1 : Générer une population initiale ($t = 0$), $P(t)$ au hasard de taille $PopSz(n, p)$.

Etape 2 : Evaluation de chaque individu de la population$P(t)$, et mémorisation de la meilleure solution.

Étape 3 : Répéter tant que ($t \leqslant MaxIter$ ou $accept \leq AC$)

Étape 4 : t=$t + 1$.

Étape 5 : Sélectionner deux solutions $P1$ et $P2$[1] de la population actuelle en utilisant une sélection de tournoi binaire.

Étape 6 : Appliquer les opérateurs génétiques

- Opérateur de croisement.
- Opérateur de mutation.

Etape 7 : Evaluation de cet nouvel ensemble d'enfants selon le type du problème à résoudre.

Etape 8 :Former la nouvelle population $P(t)$, en choisissant une partie des enfants, une partie des parents et éventuellement d'autres individus.

[1] P1,P2 : les parents

Etape 9 :Actualiser la meilleure solution.

- S'il y a amélioration incrémenter *Accept*.

    - Sinon faire $Accept = 0$ et aller à l'étape 4.

- Fin si

    Fin tant que

Fin algorithme

4.7 Exemple numérique

Nous avons programmé notre algorithme en langage *Matlab* sur un PC : Intel(R) Core(TM)2 Duo CPU T5670 @ 1,80 GHz avec 2,00 Go de mémoire de (RAM), en utilisant les 40 problèmes de " OR Library" [3]. La taille de ces 40 problèmes varie entre $n = 100$ à 900 et $p = 5$ *à* 200.

Une étude comparative avec l'algorithme génétique ADE[2] [8] (bien que les caractéristiques de l'ordinateur utilisé par[8] sont différentes du notre) est faite.

Le *Tableau* 4.1 dresse un résumé des résultats.

Tableau 4.1

[2] ADE : the algorithme génétique de Alp Osman et Erhan Erkut

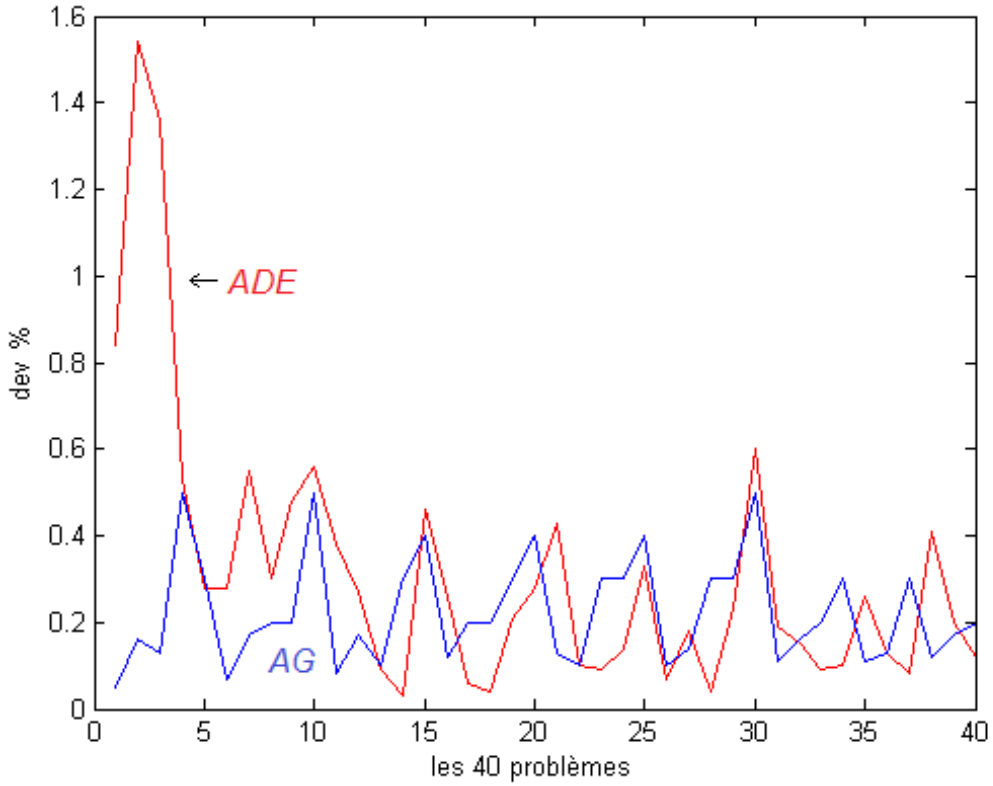
problème	n	p	*Optimal*	*Durée ADE* (sec.)[3]	*dev %* de *ADE*[4]	*Durée AG* (sec.)[5]	*dev %* de *AG*[6]
1	100	5	5819	0.1	0.84	0.12	0.05
2	100	10	4093	0.1	1.54	0.12	0.16
3	100	10	4250	0.2	1.36	0.14	0.13
4	100	20	3034	0.2	0.53	0.19	0.5
5	100	33	1355	0.3	0.28	0.13	0.3
6	200	5	8824	0.4	0.28	0.1	0.07
7	200	10	5631	0.5	0.55	0.16	0.17
8	200	20	4445	0.7	0.3	0.17	0.2
9	200	40	2734	1.2	0.48	0.2	0.2
10	200	67	1255	2.0	0.56	0.22	0.5
11	300	5	7096	1.7	0.38	0.19	0.08
12	300	10	6634	1.2	0.27	0.2	0.17
13	300	30	4374	2.1	0.09	0.3	0.1
14	300	60	2968	4.4	0.03	0.6	0.3
15	300	100	1729	6.3	0.46	0.6	0.4
16	400	5	8162	2.3	0.26	0.3	0.12
17	400	10	6999	2.4	0.06	0.3	0.2
18	400	40	4809	5.6	0.04	0.36	0.2
19	400	80	2845	13.3	0.21	0.7	0.3
20	400	133	1789	16.3	0.28	0.7	0.4
21	500	5	9138	3.8	0.43	0.2	0.13
22	500	10	8579	4.5	0.1	0.2	0.1
23	500	50	4619	15.9	0.09	0.34	0.3
24	500	100	2961	21.1	0.14	0.7	0.3
25	500	167	1828	31.6	0.33	1.53	0.4
26	600	5	9917	6.8	0.07	0.3	0.1
27	600	10	8307	7.8	0.18	0.3	0.14
28	600	60	4498	24.5	0.04	1.5	0.3
29	600	120	3033	43.7	0.23	1.5	0.3
30	600	200	1989	79.0	0.60	3.6	0.5
31	700	5	10086	14.5	0.19	0.3	0.11
32	700	10	9297	13.2	0.15	0.57	0.16
33	700	70	4700	45.4	0.09	1.2	0.2
34	700	140	3013	65.2	0.10	1.2	0.3
35	800	5	10400	15.6	0.26	2.1	0.11
36	800	10	9934	18.5	0.13	0.4	0.13
37	800	80	5057	75.9	0.08	0.7	0.3
38	900	5	11060	28.8	0.41	1.6	0.12
39	900	10	9423	26.5	0.2	0.9	0.17
40	900	90	5128	132.9	0.12	2.2	0.2

[3] le durée de déviation de l'algorithme ADE dans la seconde

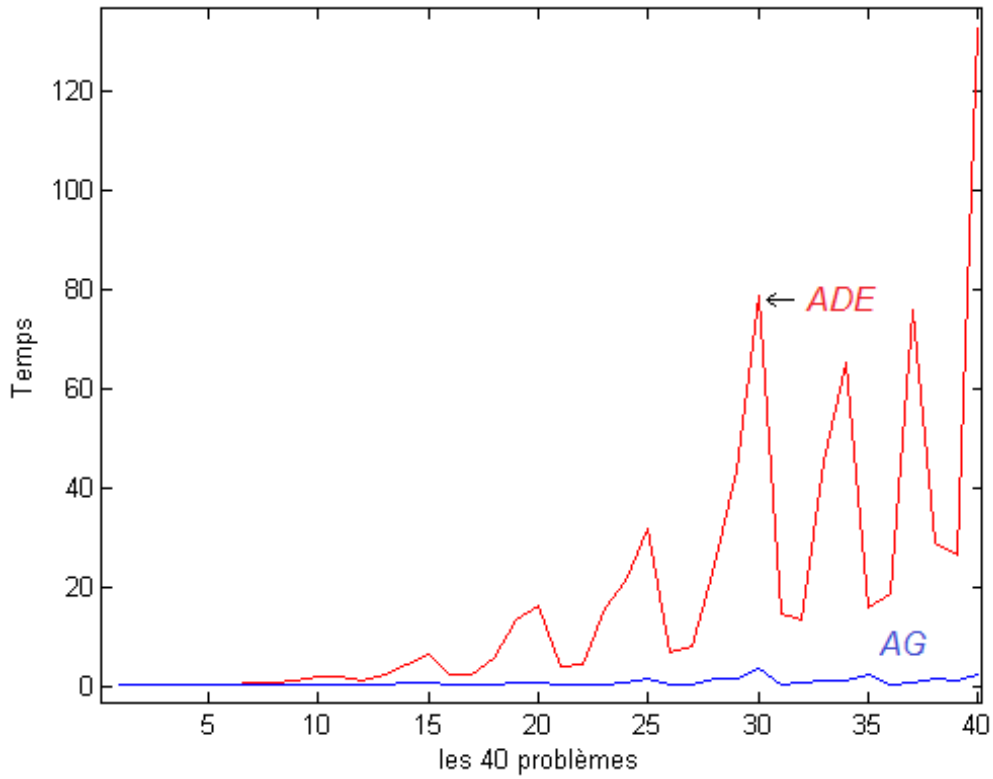
[4] Le pourcentage de déviation de l'algorithme ADE par rapport à la solution optimale.

[5] le durée de déviation de notre l'algorithme (AG) dans la seconde

[6] $dev = (\frac{sol\ obtenu\ -\ sol\ optimale}{sol\ \text{optimale}}) * 100$



[7]



[7] gfgfgfggfggf

Graphes 4.1 et 4.2

On voit bien, d'après les graphes 4.1et4.2, que nous obtenons des solutions de meilleure qualité, que ça soit en temps CPU ou en précision. Cela reste vrai surtout pour des problèmes de grande taille, ce qui est conforme à la théorie.

Conclusion Générale

Le but de notre mémoire est d'étudier le comportement des algorithmes génétiques pour la résolution d'un problème d'optimisation combinatoire tel que le probléme du P-médian.
Malgré le caractère non déterministe des algorithmes génétiques, ils restent trés efficaces et surtout pour des problèmes de grande taille. Néanmoins la présence de plusieurs paramètres (mutation , croisement , séléction , remplacement , le choix de la taille de la population, ...etc.) dans l'algorithme génétique, rend impossible la confection d'une procédure universelle pour tous les problèmes : les paramètres sont à définir pour chaque problème à traiter.
Se basant uniquement sur la bibliothèque "OR Library", comportant 40 exemples, l'algorithme génétique que nous proposons s'avère être efficace par rapport à la procédure ADE[8].
Bien que les algorithmes génétiques fournissent des solutions accéptables en un temps raisonnable, nous n'avons aucune garantie quant à la qualité de ces dernières. Il serait donc plus judicieux de combiner l'algorithme génétique avec d'autres procédures déterministes.
Nous terminons notre travail par une citation de David Goldberg et Kosorukoff , deux grands spécialistes du domaine qui ont le plus contribué à la diffusion et la vulgarisation des algorithmes génétiques :

> "La principale conclusion est que les AG ont déjà dépassé le cadre seul de leur applications dans le monde réel des systèmes vivants, ils sont prêts à entrer dans les systèmes artificiels et ils ont même déjà fait leurs premiers pas dans cette direction".
>
> (Kosorukoff et Goldberg (2001))

Bibliographie

[1] Ablaoui Hocine , "Centres et médians dans un graphe :Applications pratiques" Mémoire de magister, USTHB 1990.

[2] Abraham Jerrin George , Rajesh Kumar.S, Vineeth Mathew,Jith John Francis, Mathews Oomen, Nithin.P.Mathew,"Genetic Algorithm For Solving The Uncapacitated Facility Location Problem", International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 3, March 2013 -ISSN : 2278-0181.

[3] Beasley, J.E. (1990). "OR-Library – Distributing Test Problems by Electronic Mail." Journal of the Operational Research Society 41(11), 1069–1072.

[4] BOURAZZA Saïd , "Variantes d'algorithmes génétiques appliquées aux problèmes d'ordonnancement.

[5] Charles Darwin ,"De l'Origine des Espèces" ; PDF version Ebook ILV 1.4 (décembre 2009)

[6] C. Bontemps ."Principes mathématiques et utilisation des algorithmes génétiques ", Mimeo,(2000). Presented at the Gremaq (1995) and Lerep (1996) seminars (Toulouse).

[7] K. Shahanaghi, M. Mahmoudi, "A Genetic Algorithm For P-Median Location Problem" , International Journal of Engineering Research and Applications (IJERA) ISSN : 2248-9622 _Vol. 3, Issue 1, January -February 2013, pp.386-389.

[8] OSMAN ALP and ERHAN ERKUT "An Efficient Genetic Algorithm for the p-Median Problem". 2003 Kluwer Academic Publishers. Manufactured in The Netherlands.

[9] Mokhtar S. Bazaraa, Hanif D. Sherali , C. M. Shetty ," Nonlinear Programming : Theory and Algorithms " 3 edition .AJOHN WILEY& SONS,INC.,PUBLICATION. 2006.

[10] M.Bruno "Codage, cryptologie et applications",1^{ere}édition . 2004 $ISBN$ $2-88074-569-1$.

[11] Z. Michalewicz ,"Genetic Algorithms + Data Structures = Evolution Programs" ;3^{eme} édition .Springer, 1996