Thèse

présentée à l'Université des Sciences Sociales Toulouse I

en vue de l'obtention du doctorat

spécialité : Informatique

par

Alain Berro

Optimisation multiobjectif et stratégies d'évolution en environnement dynamique

Date de soutenance, le 18 décembre 2001

Jury composé de :

M^{me} Evelyne Lutton, Chercheur, INRIA Rocquencourt

M. Jean Marc Alliot, Professeur, CENA

M. Jean-Louis Lagouanelle, Professeur, Université Paul Sabatier

M. Claude Dupuy, Professeur, Université des Sciences Sociales

M. Jean-Pierre Jessel, Habilité à diriger des recherches, Université Paul Sabatier

M. Yves Duthen, Professeur, Université des Sciences Sociales

Université des Sciences Sociales Toulouse I, 1 place Anatole France 31042 Toulouse Cedex

Remerciements

Merci à tous.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Ces travaux de recherches s'inscrivent dans un projet regroupant des chercheurs de l'Institut de Recherche en Informatique de Toulouse (IRIT) et du Laboratoire d'Etudes et de Recherche sur l'Economie, les Politiques et les Systèmes sociaux (LEREPS). Le projet initial est de réaliser une plate-forme de simulation économique et de poser les bases de modèles génériques d'agents économiques. Ces modèles seront basés sur des techniques évolutionnaires issues de la vie artificielle. Le choix de ces techniques peut être apprécié en fonction de différents points de vue.

Les modèles économiques sont très souvent construits sur des hypothèses fortes qui limitent la plausibilité et l'utilité du modèle pour expliquer les phénomènes observés dans le monde réel. Par exemple, il est peu probable que tous les agents économiques suivent les mêmes principes de comportement. En général, un agent obéit à des règles propres, forgées par son histoire, son éducation et son apprentissage de l'environnement. Il est donc impossible d'identifier tous les stimuli externes d'un agent et les réponses à ceux-ci, comme il est également très difficile de modéliser l'ensemble des relations stimulus/réponse. La forte interaction entre agents d'un même système et la grande incertitude sur les relations de ce système empêchent une analyse totale. En économie comme dans de nombreux domaines scientifiques, l'analyse systémique est souvent découpée en deux axes : un axe global et un axe local. La macroéconomie explique le fonctionnement global d'une économie et la microéconomie explique son fonctionnement local. La macroéconomie se libère des contraintes de chaque acteur en agrégeant ou moyennant les comportements et explique l'économie par des faits généraux, des ratios... La microéconomie enferme l'acteur dans un monde restreint, sans surprise, et explique son comportement de manière systématique. L'acteur est assimilé à un automate piloté par des règles. Les techniques issues de la vie artificielle se détachent de ce découpage en essayant, à partir d'une modélisation simple des acteurs, de retrouver les modèles globaux connus, d'analyser et de comprendre l'émergence de nouveaux comportements individuels ou collectifs.

Dans une optique évolutionniste, l'avancée de la connaissance scientifique n'implique pas nécessairement un recul du hasard. L'avancée des connaissances dans la biologie a consisté à admettre la dimension positive des phénomènes aléatoires. Darwin considère les mutations comme le moteur du développement de la diversité des espèces et non pas comme un accident.

Actuellement, la naissance d'apprentis sorciers maîtrisant les techniques génétiques peut être dangereuse car si nous brisons le cercle vertueux (hasard, diversité, vie) nous risquons de condamner notre espèce. La maîtrise du hasard peut réduire la diversité et entraîner un déclin des espèces vivantes. Ce cycle est donc essentiel à tous systèmes vivants. Pour fonctionner, l'économie a également besoin de cette diversité, et la modélisation à l'aide de techniques évolutionnaires apporte tous les concepts et les outils pour introduire et maintenir la diversité dans un tel système.

Si la diversité est source de vie, elle est également source de difficultés pour la conception de modèles. En économie, la multiplicité des acteurs de par leur nombre, leur forme et leur comportement rend un problème apparemment simple très difficile à modéliser car chaque fois que l'on relâche une contrainte cela pose de nouvelles questions et remet en cause certaines hypothèses. En plus de cette diversité, la forte interaction entre les différents types d'acteurs rend la science économique difficile à modéliser par des techniques mathématiques.

Outre la diversité des acteurs et la forte interaction d'un système économique, d'autres facteurs rendent l'analyse, la modélisation et la simulation très délicates :

- Le temps : peu d'activité économique sont régies par des lois à la temporalité bien établie. Il existe une synchronisation dans les transactions mais pas dans les décisions. L'échelle de temps n'est pas la même pour un acheteur et pour une entreprise. Ce concept est d'autant plus délicat que les simulations sur ordinateur sont décrites à l'aide d'un algorithme[1].
- L'espace : le lieu de vie, de travail ou de transaction, a un impact important sur les résultats et les décisions économiques. Cette notion a été trop longtemps ignorée des modèles. A l'époque de la mondialisation, un courant de pensée actuel revalorise l'économie spatiale.
- Rationalité limitée : cette notion est en partie une conséquence des deux précédentes. La rationalité limitée est induite par un espace qu'un seul individu ne peut connaître et une capacité de mémorisation finie dans le temps. La capacité informationnelle d'un individu ou d'un groupe est inévitablement limitée par leur capacité à traiter, mémoriser et utiliser l'information disponible.
- La dynamique du système : chaque choix d'un acteur provoque un changement dans l'environnement et personne ne peut prévoir les répercutions de ce changement[2]. Par exemple, la circulation d'information est devenue tellement rapide qu'un acte isolé peut devenir symbolique pour toute une population.

[1] Méthode de résolution d'un problème utilisant un nombre fini de règles exécutées en séquence.
[2] Célèbre effet papillon du météorologue Edward Lorenz.

Ces dernières années ont vu l'émergence de techniques de vie artificielle imitant les processus naturels de l'évolution pour résoudre des problèmes complexes. Ces méthodes d'optimisation ou d'apprentissage permettent de résoudre des problèmes auxquels les méthodes classiques n'apportent pas de réponses satisfaisantes. Ces algorithmes dits évolutionnaires regroupent notamment les algorithmes génétiques [Holland 1975, Goldberg 1989], la programmation génétique [Koza 1992], les systèmes de classifieurs [Goldberg 1989] et les stratégies d'évolution [Rechenberg 1973]. L'application de ces procédés à la modélisation et la simulation d'acteurs économiques permet d'envisager de nouvelles formes de modélisation de comportements collectifs et individuels dans un environnement dynamique.

Dans mes travaux de recherche, je me suis intéressé plus particulièrement aux problèmes d'optimisation multiobjectifs et/ou en environnement dynamique. Plusieurs raisons ont motivé ce choix :

Tout d'abord, le champ d'application qui me passionne, l'économie, ne connaît pratiquement pas de problèmes à objectif unique et en environnement stable. L'économie est probablement le champ d'application le plus riche en problèmes multiobjectifs et/ou dynamiques.

L'autre raison principale est que ces deux domaines, l'optimisation multiobjectif et l'optimisation en environnement dynamique, sont actuellement en pleine progression. Même si les méthodes de résolution de problèmes multiobjectifs ne sont pas un sujet récent, celui-ci vit un renouveau depuis quatre ou cinq ans. Pour preuve, les publications sur ce sujet sont très abondantes actuellement et de nombreux domaines scientifiques tentent d'appliquer ces méthodes à leurs problèmes spécifiques. On peut attribuer le regain d'intérêt pour ces techniques à l'augmentation de la puissance des ordinateurs qui permet de simuler en temps interactif des systèmes composés de plusieurs centaines d'acteurs. Le champ de recherche sur les problèmes en environnement dynamique est en train de devenir un domaine à part entière qui étudie les mécanismes de suivi et de conservation d'un optimum.

Enfin une autre raison a été de choisir un thème de recherche peu exploré au sein de notre équipe de recherche dans le but d'apporter un nouveau point de vue pour des simulations comportementales d'agents. Les premiers travaux de simulation à travers une approche cognitive ont été introduits par Yves Duthen dans le projet IN VitrAM [Duthen 1993]. Cette approche s'est ouverte aux techniques évolutionnaires pour la création de formes et de comportement en réalité virtuelle. La thèse [Luga 1997] a présenté un état de l'art des techniques évolutionnistes et un ensemble d'applications pour la synthèse de formes et de comportements. Les recherches commencées dans mon DEA, effectué en 1995 au sein de l'équipe d'images de synthèse de l'IRIT, m'ont permis de découvrir les algorithmes génétiques comme méthode d'exploration. Le sujet était de retrouver l'équation mathématique d'un volume 3D aux courbes arrondies. Pour cela nous avons

utilisé une équation de surfaces isopotentielles comme unité de base de notre modèle géométrique et un algorithme génétique pour effectuer le paramétrage et le mélange de ces surfaces [Berro 1997]. Par la suite, Cedric Sanza a continué l'élargissement de ce champ de connaissance sur les algorithmes évolutionnaires en axant ses recherches sur les mécanismes d'apprentissage à l'aide de systèmes de classifieurs [Sanza 2001].

Après une introduction rapide sur les problèmes d'optimisation, nous présentons un état de l'art des méthodes de résolution de problèmes multiobjectifs. Ces méthodes sont classées en trois catégories. Après une brève introduction théorique, nous présentons les méthodes le plus usités de chaque catégorie. Nous terminons ce chapitre en présentant une synthèse des caractéristiques principales de ces méthodes et de leurs difficultés d'utilisation et de mise en œuvre.

Ensuite nous présentons un état de l'art des méthodes d'optimisation en environnement dynamique. Les méthodes sont classées en fonction de leurs approches de détection et de réponse à un changement d'environnement. Dans notre synthèse sont mises au premier plan leur capacité de détection endogène ou exogène d'un changement et leur approche discriminante ou non discriminante de l'espace de recherche.

Dans une deuxième partie nous développons, pour la résolution de problèmes multimodaux[3] en environnement dynamique, une méthode d'optimisation multiagent utilisant des mécanismes issus des stratégies d'évolution. La plupart des méthodes en environnement dynamique présentées dans l'état de l'art ont une approche non discriminante de l'espace de recherche. Dès lors, le temps de réponse au changement de l'environnement et le risque de détruire des niches situées dans d'autres régions de l'espace sont augmentés. Les apports originaux de notre méthode sont l'introduction de la notion de zone d'influence et l'utilisation d'un facteur de précision pour rechercher les optima. Ces caractéristiques procurent à notre méthode une forte capacité discriminante de l'espace de recherche et une capacité endogène de détection du changement. Lors de la conception de cette méthode, nous nous sommes également attachés à réduire le nombre de paramètres de réglage et à rendre ces paramètres intelligibles pour l'utilisateur.

Nous présentons ensuite une généralisation de cette méthode à des problèmes multiobjectifs en utilisant la notion de dominance au sens de Pareto. Nous redéfinissons la notion de zone d'influence pour ce type de problème et nous proposons une interprétation de cette notion.

La troisième partie est consacrée à la présentation de deux applications des techniques évolutionnaires à des problèmes économiques.

Nous terminons par la présentation des perspectives ouvertes par ces travaux et les futures directions de recherches envisagées.

[3] Un problème multimodal est un cas particulier de problème multiobjectif.

Partie I.

Les problèmes d'optimisation

Chapitre 1.
Introduction

Les problèmes d'optimisation occupent actuellement une place de choix dans la communauté scientifique. Non pas qu'ils aient été un jour considérés comme secondaires mais l'évolution des techniques informatiques a permis de dynamiser les recherches dans ce domaine.

Le monde réel offre un ensemble très divers de problèmes d'optimisation :

- problème combinatoire ou à variables continues,
- problème à un ou plusieurs objectif(s),
- problème statique ou dynamique,
- problème dans l'incertain.

Cette liste n'est évidemment pas exhaustive, et un problème peut être à la fois multiobjectif et dynamique.

Dans les paragraphes suivants, nous n'aborderons que les problèmes à variables continues. Pour les problèmes combinatoires le lecteur pourra se référer à [Ehrgott and Gandibleux 2000].

1.1 Définitions

Un problème d'optimisation est défini par un espace d'état[4], une ou plusieurs fonction(s) objectif(s)[5] et un ensemble de contraintes[6].

L'espace d'état est défini par l'ensemble des domaines de définition des variables du problème. Dans la plupart des problèmes, cet espace est fini car la méthode de résolution utilisée a besoin de travailler dans un espace restreint (Exemples : la méthode Monte-Carlo, les algorithmes

[4] Syn. Espace de recherche des solutions.
[5] Syn. Critère.
[6] Syn. Condition.

génétiques). Cette limitation n'est pas problématique car lorsqu'un problème est posé, le décideur[7] précise un domaine[8] de valeurs envisageable à chacune des variables. De plus, pour des raisons opératoires[9] et de temps de calcul, il est préférable de travailler sur des domaines finis.

Les variables du problème peuvent être de nature diverse (réelle, entière, booléenne, etc.) et exprimer des données qualitatives ou quantitatives. Nous ne développerons pas ce thème mais le paragraphe 2.5.1.3 montre la difficulté de mise en œuvre de certaines méthodes si les variables sont de types différents.

Une fonction objectif représente le but à atteindre pour le décideur (minimisation de coût, de durée, d'erreur, etc.). Elle définit un espace de solutions potentielles au problème.

L'ensemble de contraintes définit des conditions sur l'espace d'état que les variables doivent satisfaire. Ces contraintes sont souvent des contraintes d'inégalité ou d'égalité et permettent en général de limiter l'espace de recherche.

La séparation entre les fonctions objectifs et les contraintes peut paraître artificielle car nous pourrions considérer qu'une contrainte est un objectif à atteindre. Mais elle se justifie de deux manières différentes. D'une part, les contraintes sont appliquées sur l'espace de recherche alors que les objectifs définissent l'espace des solutions. D'autre part, dans de nombreuses méthodes les contraintes et les objectifs sont traités par des procédures différentes.

Une méthode d'optimisation recherche le point[10] ou un ensemble de points de l'espace des états possibles qui satisfait au mieux un ou plusieurs critère(s). Le résultat est appelé *valeur optimale* ou *optimum*.

1.2 Choix d'une méthode

La nature des variables, des domaines de définition et des critères à optimiser va influencer le choix de la méthode d'optimisation à utiliser.

Il y a deux grandes familles de méthodes d'optimisation : les méthodes déterministes et les méthodes stochastiques.

1.3 Les différentes méthodes

La manière la plus naturelle et la plus ancienne de résoudre un problème d'optimisation est la méthode par essai/erreur. Le décideur corrige ses actions en fonction des résultats jusqu'à obtenir

[7] Le terme de décideur sera souvent utilisé pour identifier l'utilisateur d'une méthode.
[8] La détermination d'un domaine vient de l'expérience du décideur ou du simple bon sens.
[9] Un domaine infini de valeurs n'est pas représentable en code binaire.

une solution satisfaisante. Cette méthode, apparemment simpliste, est à la base d'un très grand nombre de méthodes d'optimisation.

1.3.1 Les méthodes déterministes

1.3.1.1 Définition

Les méthodes déterministes se caractérisent par une exploration systématique[11] de l'espace de recherche.

Il existe de nombreuses méthodes d'optimisation déterministes. **Les méthodes locales** qui assurent la convergence vers l'optimum de la fonction le plus proche de la solution courante en explorant son voisinage et **les méthodes globales** qui s'attachent à faire converger la solution vers l'optimum global de la fonction.

Nous allons présenter les méthodes de gradient, la méthode multistart et brièvement la méthode du simplexe qui sont des méthodes de recherche locale. D'une manière générale ces méthodes obéissent à l'algorithme suivant :

a) Choix d'une première solution courante *i* admissible

b) Génération d'une solution *j* dans le voisinage de *i*

c) Si *f*(*j*) est meilleur que *f*(*i*) alors *j* devient la solution courante puis retour en b)

d) L'algorithme se termine lorsqu'il n'y a plus d'amélioration de la solution courante

Ensuite nous présentons les méthodes de recherche globale que sont la méthode de séparation - évaluation (branch and bound) et la méthode de tunneling. Les méthodes homotopiques, l'algorithme A^* et la programmation linéaire qui sont également des méthodes d'optimisation globale ne seront pas présentés.

1.3.1.2 Les méthodes de gradient

Historiquement, les méthodes de gradient sont les plus anciennes. Elles permettent de résoudre des problèmes non linéaires [Minoux 1983] et sont basées sur une hypothèse forte : la connaissance de la dérivée de la fonction objectif en chacun des points de l'espace. Cette famille de méthodes procède de la façon suivante :

[10] Dans le cadre de problèmes d'aide à la décision, un point est appelé *action* et un optimum est *une action efficace*.
[11] Syn. Méthodique.

On choisit un point de départ x_0 et on calcule le gradient $\nabla f(x_0)$ en x_0. Comme le gradient indique la direction de plus grande augmentation de f, on se déplace d'une quantité λ_0 dans le sens opposé au gradient et on définit le point x_1 :

$$x_1 = x_0 - \lambda_0 \frac{\nabla f(x_0)}{\|\nabla f(x_0)\|} \qquad 1)$$

Cette procédure est répétée et engendre les points $x_0, x_1, \ldots, x_k$. Ainsi, pas à pas, la distance entre le point d'indice k et l'optimum diminue.

$$x_{k+1} = x_k - \lambda_k \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|} \text{ où } \forall k, \lambda_k > 0 \qquad 2)$$

λ_k est le pas de déplacement à chaque itération.

Si λ_k est fixé, on parle de **méthode de gradient à pas prédéterminé**. L'inconvénient de cette procédure est que la convergence est très dépendante du choix du pas de déplacement. La convergence peut être très lente si le pas est mal choisi[12]. L'intérêt principal de cette méthode est de pouvoir se généraliser aux cas de fonctions non partout différentiables.

Actuellement, la méthode la plus usitée de cette famille est la **méthode de la plus forte pente**. Elle permet de se libérer du choix d'un λ_k mais elle introduit un critère d'arrêt. Le but de cette méthode est de minimiser la fonction de λ :

$$g(\lambda) = f(x_k - \lambda.\nabla f(x_k)) \qquad 3)$$

Algorithme de la plus forte pente

a) Choisir un point de départ x_0 et faire $k = 0$.

b) A l'itération k : $d_k = -\nabla f(x_k)$.

Recherche λ_k tel que : $f(x_k + \lambda_k.d_k) = \text{Min}\{ f(x_k + \lambda.d_k) \}$ pour $\lambda \geq 0$.

$x_{k+1} = x_k + \lambda_k.d_k$.

c) **Si** le test d'arrêt est vérifié **alors** fin **sinon** $k \leftarrow k + 1$ et retourner en b)

L'algorithme de la plus forte pente est à la base de l'algorithme de Hill-Climbing appelé également algorithme de descente de gradient.

[12] Pour plus de détails, on se reportera au paragraphe 3 chapitre 4 de [Minoux 1983].

Difficultés

Le défaut majeur de cette méthode est que la convergence peut être ralentie dans certains types de fonctions.

- Si les déplacements sont optimaux, on en déduit que deux directions de déplacements successifs sont orthogonales [Minoux 1983, p 103]. Donc des fonctions de type *vallée étroite* ou *allongée* vont piéger les x_k et ralentir la convergence.
- Si la fonction est convexe[13] alors ces algorithmes d'optimisation convergent vers l'optimum global. Mais dans le cas d'une fonction non convexe, ces méthodes risquent de converger vers un optimum local dépendant de la valeur du point de départ x_0 du processus de recherche.
- Si la fonction présente des zones plates, la convergence sera ralentie.
- Si la fonction présente des zones raides, la recherche va être fortement influencée par ces pentes plus que par la progression vers le sommet.

Une manière d'éviter ces problèmes est d'utiliser la méthode multistart.

1.3.1.3 La méthode multistart

La méthode multistart effectue une recherche locale à partir de plusieurs points répartis dans l'espace de recherche. Avant de lancer le processus de recherche, il faut réaliser un maillage de l'espace de recherche. L'efficacité de cette méthode dépend de la bonne adéquation entre le maillage et la forme de la fonction objectif. Si le maillage est trop grand, la probabilité de trouver l'optimum global sera faible car certaines vallées[14] ne seront pas traitées. Si le maillage est trop petit, la recherche globale sera inefficace car plusieurs points vont être présents dans la même vallée et convergeront donc vers le même optimum.

Pour éviter ce dernier inconvénient, la méthode a été améliorée en introduisant la notion de cluster[15]. Le regroupement des points voisins permet de diminuer le nombre de calculs en évitant les recherches locales redondantes. Les points à partir desquels la recherche locale est relancée étant choisis plus minutieusement dans chaque cluster, l'efficacité de l'algorithme est ainsi augmentée.

Difficultés

Les méthodes multistart sont efficaces dans de nombreux problèmes et sont simples à mettre en œuvre. L'inconvénient principal est le choix du maillage de l'espace de recherche qui doit être

[13] Une fonction $f : R^n \rightarrow R$ est convexe si elle vérifie $\forall x, y \in R^n$ et $\forall a \in [0, 1], f(a.x + (1\text{-}a).y) \leq a.f(x) + (1\text{-}a).f(y)$.
[14] Le terme de vallée est utilisée car nous considérons que nous sommes dans un problème de minimisation. Si nous étions dans un problème de maximisation, nous utiliserions le terme de pic.
[15] Une technique de création de clusters dans le cadre de problèmes multiobjectifs est présentée en annexe.

suffisamment précis pour être efficace, sinon la convergence vers l'optimum global ne peut pas être assurée. L'alternative pour s'exempter du maillage est de générer les points de manière aléatoire. Les méthodes utilisant cette technique sont nommées **méthodes random multistart**.

Critique

L'inconvénient majeur des deux types de méthodes présentés ci-dessus est leur hypothèse de dérivabilité qui les rend inaptes à traiter beaucoup de problèmes réels. De plus, leur déplacement déterministe dans l'espace d'état engendre des temps de calcul très importants dans certaines conditions :

- espace d'état très grand,
- mauvais paramétrage de la méthode (maillage, valeur du pas de déplacement),
- piège de certaines fonctions.

1.3.1.4 La méthode de Nelder Mead ou la méthode du simplexe

L'intérêt principal de la méthode du simplexe par rapport aux deux précédentes est qu'elle ne nécessite pas de calcul de gradient. Elle est uniquement basée sur l'évaluation de la fonction. Cela la rend utilisable pour des fonctions bruitées.

Cette méthode locale effectue une recherche multidirectionnelle dans l'espace d'état [Nelder and Mead 1965]. Soit une fonction f à minimiser, on appelle **simplexe** de R^n tout ensemble (x_0, x_1, …, x_n) de points de R tel que $f(x_0) \geq f(x_i)\ \forall\ i \in [1..n]$. x_0 est donc le meilleur élément.

Après la construction d'un simplexe initial, l'algorithme va modifier celui-ci en appliquant des calculs de réflexions, expansions et contractions jusqu'à la validation d'un critère d'arrêt.

1.3.1.5 L'algorithme de séparation - évaluation : branch and bound

Le principe de cette méthode est de découper (branch) l'espace initial de recherche en domaines de plus en plus restreints afin d'isoler l'optimum global (principe de séparation). L'algorithme de recherche forme ainsi un arbre dont chaque nœud représente une partie de l'espace. Ensuite chaque nœud est évalué de façon à déterminer sa borne (bound) inférieure[16] en fonction d'un critère d'évaluation. Si cette borne n'est pas meilleure que la solution courante alors la recherche sur ce nœud est stoppée, sinon la séparation continue.

Critique

L'efficacité de cette technique dépend essentiellement du choix des critères de séparation et d'évaluation. Un bon choix permettra à l'algorithme de réduire l'arbre de recherche en évitant la construction de certaines branches. Dans le pire des cas, un mauvais choix peut amener

[16] Inférieure dans le cas d'une minimisation, supérieure dans le cas d'une maximisation.

l'algorithme à explorer tout l'espace de recherche. L'utilisation de cette méthode reste limitée à des espaces de petites dimensions.

Cette méthode est essentiellement utilisée sur des problèmes discrets. Couplée avec une méthode d'analyse d'intervalle [Messine 1997] elle peut être également utilisée dans le cadre de problèmes continus avec l'avantage de ne pas exiger la continuité de la fonction et d'éviter que la propagation d'erreurs numériques empêche la validation d'une solution.

1.3.1.6 La méthode du "Tunneling"

Cette méthode recherche l'optimum global d'une fonction en effectuant des recherches successives d'optima locaux telles que la valeur de la fonction s'améliore. L'algorithme se décompose en deux phases : une phase de recherche d'un optimum local et une phase dite de tunneling.

Si on se place dans le cas d'une fonction f à minimiser, la première phase peut utiliser une méthode de gradient pour trouver le minimum local $f^* = f(x^*)$.

La deuxième phase consiste à trouver un point x dans une autre vallée à l'aide d'une fonction T de tunneling du type : $T(x) = f(x) - f^* \leq 0$

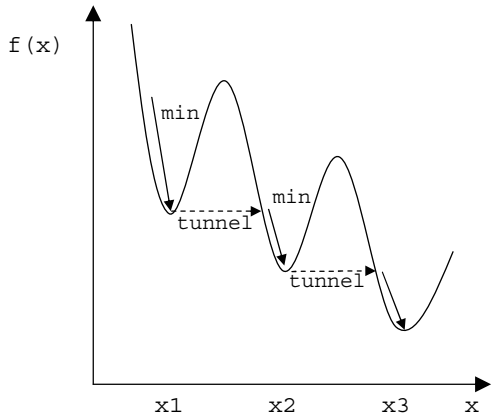


Figure 1 : Méthode du "Tunneling"

Critique

Le principal inconvénient de cette méthode est l'élaboration de la fonction de tunneling.

1.3.1.7 Conclusion

Les méthodes déterministes vues ci-dessus sont très efficaces sur des problèmes particuliers et en général de petite taille. Mais sur des problèmes de grande taille, la probabilité de trouver l'optimum global en un temps raisonnable dépend essentiellement de la bonne connaissance du problème par le décideur. Ainsi, ce dernier pourra choisir avec précision la bonne méthode et les bons paramètres.

Si les conditions exposées ci-dessus ne sont pas réunies, le décideur devra plutôt s'orienter vers des méthodes stochastiques.

1.3.2 Les méthodes stochastiques

1.3.2.1 Définition

Les méthodes stochastiques sont caractérisées par :

- un processus de création aléatoire ou pseudo-aléatoire[17] des points dans l'espace d'état,
- une heuristique qui permet de guider la convergence de l'algorithme.

Ces méthodes sont utilisées dans des problèmes où on ne connaît pas d'algorithme de résolution en temps polynomial et pour lesquels on espère trouver une solution approchée de l'optimum global.

Dans la pratique, les méthodes stochastiques qui connaissent le plus de succès sont : la méthode Monte Carlo, le recuit simulé, les algorithmes évolutionnaires, le branch and bound stochastique et la méthode tabou. Leurs atouts principaux sont les suivants :

- facilité d'implantation,
- flexibilité par rapport aux contraintes des problèmes,
- qualité élevée des solutions.

D'un point de vue théorique, il existe des théorèmes de convergence pour les algorithmes génétiques [Goldberg 1989, Raphaël Cerf 1994] et pour le recuit simulé [Hajek 1988] qui justifient l'usage de ces méthodes. En général, il est établi que l'on a une probabilité très élevée de trouver une solution optimale, si un temps de calcul très important est alloué.

1.3.2.2 La méthode Monte Carlo

Les méthodes de type Monte Carlo recherchent l'optimum d'une fonction en générant une suite aléatoire de nombres en fonction d'une loi uniforme [Fishman 1997].

Algorithme

a) On génère un point initial x dans l'espace d'état, considéré comme solution courante.

b) On génère aléatoirement un point x'.

c) Si x' est meilleur que x alors x' devient la solution courante.

d) Si le critère d'arrêt est satisfait alors fin sinon retour en b)

1.3.2.3 Le recuit simulé

La méthode du recuit simulé est basée sur une analogie avec le processus physique de recuit des matériaux cristallins. Ce dernier consiste à amener un solide à basse température après l'avoir élevé à forte température. Lorsque le solide est à une forte température, chaque particule possède une très

[17] La création des points est aléatoire mais en la conditionnant dans un intervalle de valeurs, la création devient guidée.

grande énergie et peut effectuer de grands déplacements aléatoires dans la matière. Au fur à mesure que la température est abaissée, chaque particule perd de l'énergie et sa capacité de déplacement se réduit. Les différents états transitoires de refroidissement permettent d'obtenir des matériaux très homogènes et de bonne qualité.

Pour appliquer ce comportement à une méthode d'optimisation, les déplacements aléatoires de chacun des points vont être liés à une probabilité dépendante d'une variable T représentant la température du matériau.

Algorithme

a) A partir d'un point initial x_0, on effectue un déplacement aléatoire (changement d'état).

b) Si le déplacement mène à x_1 tel que $f(x_1) < f(x_0)$ alors x_1 est accepté.

Sinon, il est accepté avec une probabilité $p = exp(-|\Delta(f)| / kT)$.

$\Delta(f)$ représente la distance de déplacement : $x_1 - x_0$.

T est assimilé à une température décroissante au cours du temps.

k est une constante.

Au début de la simulation, les points ont une grande capacité d'exploration de l'espace d'état car l'algorithme accepte des déplacements très importants. De nombreux points n'améliorant pas leur valeur dans f sont acceptés car la probabilité p est grande. Au fur et à mesure que T diminue (p augmente), la capacité de déplacement d'un point diminue et les points améliorant leur valeur sont de plus en plus nombreux. Quand $T \rightarrow 0$, seuls les points améliorant leur valeur sont acceptés.

Schéma de fonctionnement

Supposons une bille qui glisse le long d'une surface. Si cette dernière est convexe la bille va atteindre le point minimal de la surface après plusieurs oscillations.

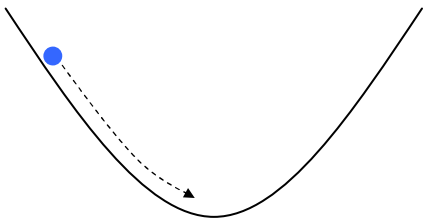


Figure 2 : Fonction convexe

Si cette surface n'est pas convexe, c'est-à-dire si elle possède au moins un optimum local, la bille risque de se bloquer dans un optimum (Figure ci-dessous) si son énergie de départ n'est pas assez importante. Avec une énergie initiale importante, la bille pourra éviter le piège.

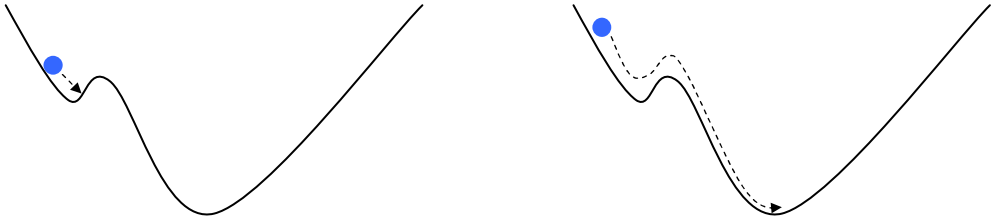


Figure 3 : Fonction non convexe

Le fait de partir d'un niveau de température élevée donne à l'algorithme une bonne capacité d'exploration et un refroidissement assez lent évite que la recherche s'arrête sur un minimum local.

1.3.2.4 Les algorithmes évolutionnaires

Les algorithmes évolutionnaires sont inspirés des concepts issus du Lamarckisme, Darwinisme et du mutationnisme. Un historique sur ces recherches est présenté en annexe (p. 162).

Les algorithmes évolutionnaires se caractérisent par :

- Une représentation spécifique des solutions potentielles du problème.
- Un ensemble d'individus formant une population permettant de mémoriser les résultats à chaque étape du processus de recherche.
- Un processus de création aléatoire d'un individu. Cette caractéristique offre une capacité exploratoire importante à la méthode.
- Un ensemble d'opérateurs de modification permettant de créer de nouvelles solutions à partir des informations mémorisées. Ces opérateurs offrent une capacité de recherche locale à la méthode.
- Une heuristique de notation qui représente la sélection effectuée par l'environnement.
- Une heuristique de sélection.
- Un critère d'arrêt de l'algorithme.

On distingue plusieurs types d'algorithmes évolutionnaires.

Les algorithmes génétiques [Holland 1975, Goldberg 1989] sont inspirés des mécanismes de l'évolution naturelle. Une description est présentée en annexe (p. 162).

La programmation génétique est une extension des algorithmes génétiques dans laquelle les individus sont des programmes. Le génotype d'un individu est constitué d'un alphabet et se présente sous forme arborescente [Koza 1992].

Les systèmes de classifieurs sont des mécanismes d'apprentissage basés sur un ensemble de règles condition/action. Chaque règle est notée en fonction du résultat de l'action produite et un algorithme génétique est utilisé pour générer de nouvelles règles [Goldberg 1989].

Les stratégies d'évolution [Rechenberg 1973] sont des algorithmes itératifs dans lesquels un parent génère un enfant (*1*+*1*)-ES. Le meilleur des deux survit et devient le parent de la génération suivante. La généralisation de ce processus a donné les algorithmes (μ+λ)-ES dans lesquels μ parents génèrent λ enfants. Les μ meilleurs survivent.

Avantages

- Ces méthodes sont applicables dans la plupart des problèmes d'optimisation : multimodaux, non continus, contraints, bruités, multiobjectifs, dynamiques, etc.
- Elles n'exigent pas d'hypothèse par rapport à l'espace d'état.
- Elles sont flexibles par rapport aux nouvelles contraintes et nouveaux critères à prendre en compte.
- Les résultats sont en général exploitables et interprétables par le décideur.

Inconvénients

- Elles n'offrent aucune garantie de trouver l'optimum en un temps fini. Mais cela est vrai pour toutes les méthodes d'optimisation globales.
- Leur base théorique est insuffisante.
- Le réglage des paramètres est largement inspiré du essai/erreur sauf pour les stratégies d'évolution qui sont auto-adaptatives.

1.3.2.5 Le branch and bound stochastique

Cette méthode utilise le même principe que le branch and bound déterministe mais la borne inférieure (ou supérieure) est stochastique. Elle apparaît plus simple à paramétrer et elle peut être utilisée dans des espaces de recherche plus grands.

1.3.2.6 La méthode Tabou

La recherche Tabou [Glover and Laguna 1997] est une méthode de recherche locale. A chaque itération, l'algorithme examine le voisinage V d'un point x et retient le meilleur voisin x' tel que $\forall$ $x'' \in V(x), f(x')$ est meilleur que $f(x'')$ et $x' \notin$ à la liste tabou, sauf si $f(x')$ est meilleur que $f(x)$: c'est le phénomène d'aspiration. On note que x' est retenu même si $f(x')$ n'est pas meilleur que $f(x)$.

L'élément fondamental de la méthode tabou est l'utilisation d'une mémoire dynamique dite liste tabou qui permet d'enregistrer les informations pertinentes des étapes de recherche précédentes. Cette liste empêche le blocage de la recherche sur un optimum local en interdisant à plus ou moins court terme de revenir sur des solutions précédemment visitées de l'espace d'état, solutions dites taboues. La durée de cette interdiction dépend d'un paramètre k appelé teneur tabou. Ce dernier est

difficile à déterminer[18] car il est très dépendant de la nature du problème. Si la valeur est faible la méthode risque de se bloquer sur un optimum local alors qu'une valeur élevée diminuera la capacité de la méthode à exploiter le voisinage de la solution courante.

La méthode Tabou est souvent utilisée avec des techniques dites d'intensification et de diversification. L'intensification est fondée sur l'enregistrement des propriétés des situations a priori de bonne qualité afin de les exploiter ultérieurement. La diversification cherche à diriger l'algorithme vers des régions de l'espace de recherche non encore explorées.

1.4 Discussion : compromis entre exploration et exploitation

La **capacité d'exploitation** est l'aptitude d'une méthode à utiliser des résultats déjà obtenus pour faire converger l'algorithme.

La **capacité d'exploration** est l'aptitude d'une méthode à explorer avec efficacité l'espace d'état. Cette aptitude a tendance à ralentir la convergence.

Lors de la conception d'une méthode d'optimisation, le chercheur doit choisir entre maintien de la diversité et convergence alors que le décideur doit choisir entre *efficacité* et *efficience* d'une méthode.

Une méthode efficace fournira un résultat optimal ou proche de l'optimum. Dans ce cas, le temps de calcul n'a aucune importance. Pour assurer un résultat optimal, ces méthodes sont obligées d'effectuer une recherche exploratoire très importante de façon à ne laisser aucune partie de l'espace des états non visitée. Ces méthodes optent pour un maintien de la diversité dans le temps. Or, si l'on fait l'hypothèse d'une population de taille fixe, ce maintien entraîne un ralentissement de la convergence.

Une méthode efficiente est une méthode capable de donner un résultat satisfaisant en un temps de calcul relativement court[19]. Dans ce genre de méthode, le décideur privilégie le temps de calcul à la précision du résultat. La conception de ces méthodes est basée sur une capacité d'exploitation importante des résultats précédents. Ainsi, le processus de convergence est accéléré au risque de voir la méthode trompée et dirigée dans une zone de l'espace non optimale. Cette réutilisation systématiquement des caractères des générations passées entraîne une perte rapide de diversité. L'adaptation des individus n'a pas le temps de devenir optimale.

Les méthodes Monte Carlo ont une bonne capacité d'exploration mais une mauvaise capacité d'exploitation. Dans les méthodes de gradient ou multistart, l'exploitation des résultats précédents est efficace mais leur capacité d'exploration est limitée. Les méthodes de recuit simulé et les

[18] Peut être fixé de façon statique ou dynamique.

algorithmes génétiques offrent pour certains types de problèmes un bon compromis entre exploration et exploitation qui dépend du choix judicieux des paramètres de réglages.

Après cette introduction synthétique sur les problèmes d'optimisation, nous allons nous intéresser plus particulièrement aux problèmes d'optimisation multiobjectifs.

[19] La durée du calcul est à apprécier au regard du type de problème traité.

Chapitre 2.
Les problèmes d'optimisation multiobjectifs

2.1 Introduction

Dans ce chapitre, nous présentons tout d'abord un ensemble de définitions liées aux problèmes d'optimisation multiobjectif. Ensuite, nous exposons la problématique issue de ces problèmes et les deux types de classification des méthodes de résolution. Pour terminer, nous présentons un large éventail de ces méthodes en essayant d'apporter un regard critique sur chacune d'elles.

2.2 Définitions

2.2.1 Problème multiobjectif

Un problème multiobjectif ou **multicritère** peut être défini comme un problème dont on recherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectifs.

Par la suite, nous allons voir que les problèmes d'optimisation ont en général plusieurs solutions car la définition d'un optimum ne peut pas être établie dans les problèmes multiobjectifs.

Le paragraphe suivant donne la définition mathématique d'un problème d'optimisation multiobjectif. Nous conserverons les mêmes notations dans la suite du document.

Définition

Une action (ou **un vecteur de décisions)** sera notée :

$$x = (x_1, x_2, \ldots, x_n) \qquad 4)$$

avec x_i les variables du problème et n le nombre de variables.

Les contraintes[20] seront notées :

$$g_i(x) \text{ avec } i = 1, ..., m \qquad 5)$$

avec m le nombre de contraintes.

Le vecteur de fonctions objectifs sera noté f :

$$f(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \qquad 6)$$

avec f_i les objectifs ou critères de décision et k le nombre d'objectifs.

Nous considérons que les objectifs sont des fonctions de minimisation[21].

Un problème d'optimisation recherche l'action x^* telle que les contraintes $g_i(x^*)$ soient satisfaites pour $i = 1, ..., m$ et qui optimise la fonction f : $f(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_k(x^*))$

L'union des domaines de définition de chaque variable et les contraintes définies en 5) forment un ensemble E que nous appelons **l'ensemble des actions réalisables**.

Nous appellerons F **l'ensemble des objectifs réalisables**.

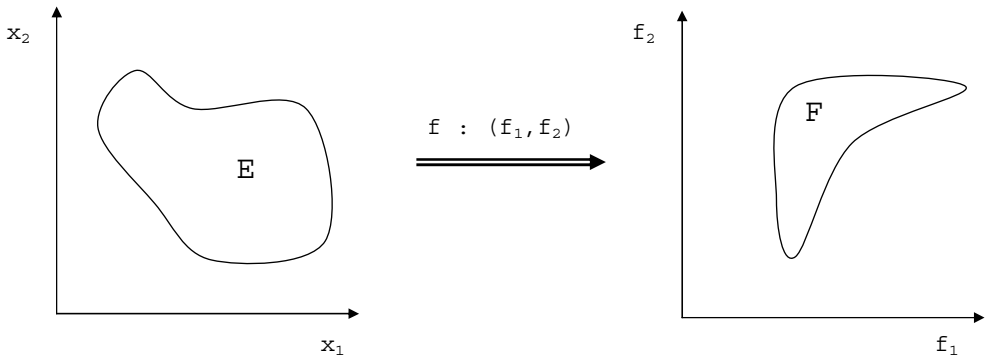


Figure 4 : Définition de E, F et f.

2.2.2 Le vecteur idéal

Le vecteur idéal est l'action composée de la solution optimale pour chaque objectif pris séparément.

$$x^0 = (x_1^0, \ldots, x_n^0) \qquad 7)$$

avec x_i^0 la valeur qui optimise la i^{ème} fonction objectif.

La condition nécessaire et suffisante pour que ce vecteur idéal soit atteint est que les fonctions objectifs soient indépendantes. Si cette condition est réalisée alors la résolution du problème multiobjectif est transformée en une résolution de plusieurs problèmes uniobjectifs.

[20] Contraintes d'égalité ou d'inégalité.
[21] Dans le cas de fonction f de maximisation, il suffit de minimiser $-f$.

2.2.3 Convexité

L'ensemble *F* est dit convexe si tout segment joignant deux points quelconques de *F* est inclus dans *F*.

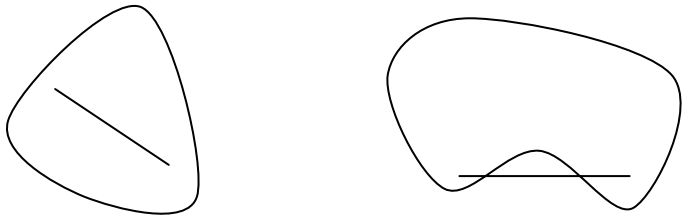


Figure 5 : Espace convexe (à gauche) et non convexe (à droite)

2.3 Problématique

La difficulté principale d'un problème multiobjectif est qu'il n'existe pas de définition de la solution optimale. Le décideur peut simplement exprimer le fait qu'une solution est préférable à une autre mais il n'existe pas une solution meilleure que toutes les autres.

Dès lors résoudre un problème multiobjectif ne consiste pas à rechercher la solution optimale mais l'ensemble des solutions satisfaisantes pour lesquelles on ne pourra pas effectuer une opération de classement. Les méthodes de résolution de problèmes multiobjectifs sont donc des méthodes d'aide à la décision car le choix final sera laissé au décideur.

Pour répondre à ce problème la communauté scientifique a adopté deux types de comportement. Le premier est de ramener un problème multiobjectif à un problème simple objectif au risque d'enlever toute signification au problème. Le second comportement est de tenter d'apporter des réponses au problème au prenant en compte l'ensemble des critères. Cette partie de la communauté scientifique a amené durant ces dix dernières années un grand nombre d'innovations dans les méthodes de résolution. La différence entre ces deux communautés s'exprime dans le schéma ci-dessous. Soit le décideur intervient dès le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectif en un problème simple objectif. Soit le décideur effectue son choix dans l'ensemble des solutions proposées par le solveur multiobjectif.

La principale qualité d'un solveur multiobjectif est donc de rendre les décisions plus faciles et moins subjectives en proposant un sous-ensemble représentatif de *F*.

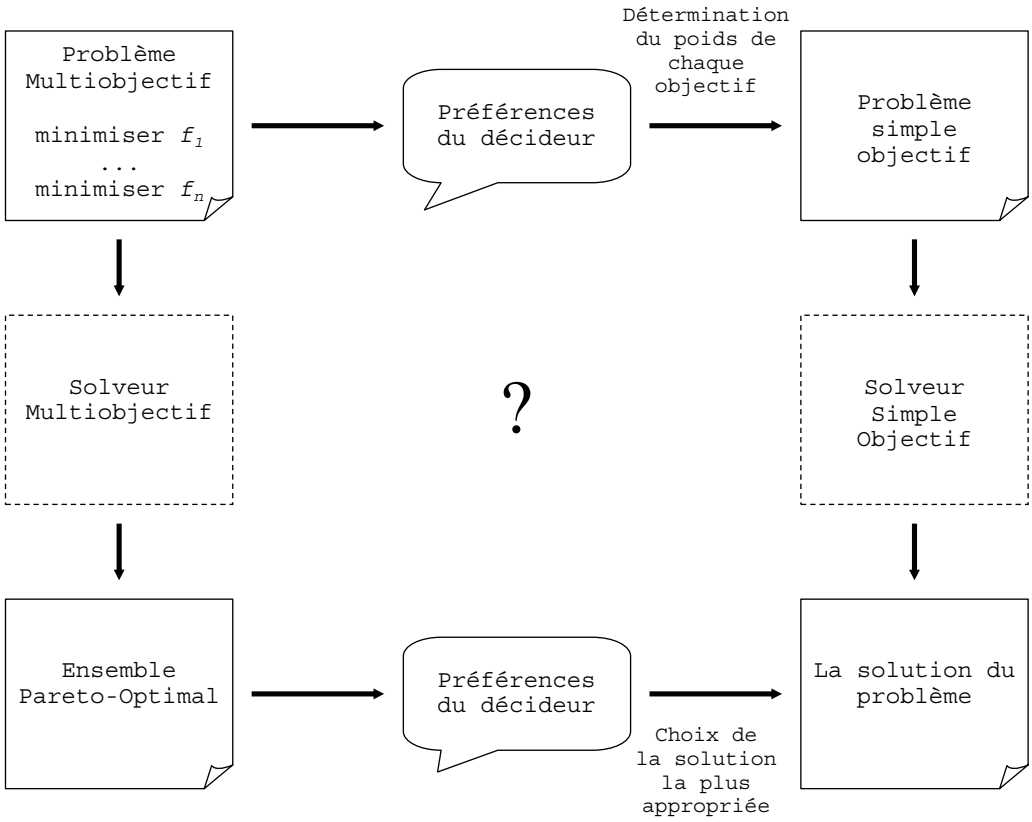


Figure 6 : Quel mode de résolution choisir ?

Supposons que l'on souhaite minimiser deux fonctions. La figure ci-dessous présente l'espace des objectifs réalisables.

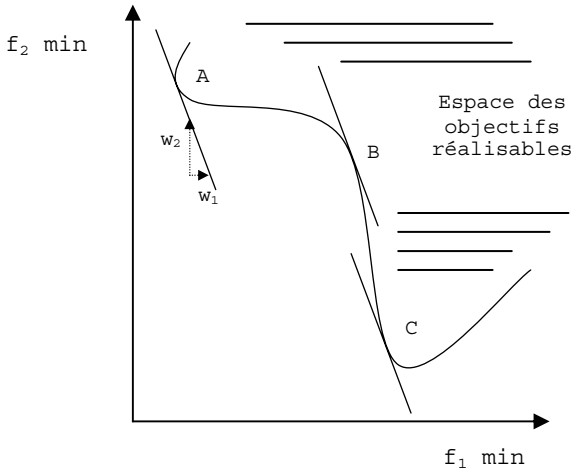


Figure 7 : Problématique

Si le décideur opte pour une méthode agrégée avec w_1 et w_2 comme poids des objectifs alors le solveur simple objectif va faire converger la solution vers les solutions *A* ou *C*. Or toutes les solutions sur la portion de courbe entre *A* et *C* peuvent également satisfaire le décideur.

L'utilisation d'un solveur multiobjectif permet d'obtenir un ensemble de points situés entre A et C donnant ainsi au décideur une plus vaste gamme d'actions.

2.4 Classification

Dans les différentes publications, nous rencontrons deux classifications différentes des méthodes de résolution de problèmes multiobjectifs. Le premier classement adopte un point de vue utilisateur[22], les méthodes sont classées en fonction de l'usage que l'on désire en faire. Le deuxième classement est plus théorique, plus conceptuel, les méthodes sont triées en fonction de leur façon de traiter les fonctions objectifs.

2.4.1 Utilisateur

Cette classification est essentiellement utilisée en recherche opérationnelle. Les décisions étant considérées comme un compromis entre les objectifs et les choix spécifiques du décideur (contraintes de coût, de temps, etc.), un décideur choisit une méthode en fonction de l'aide qu'elle va lui apporter.

Les méthodes a priori (décideur → recherche)

Les solutions les plus intuitives pour résoudre des problèmes multiobjectifs consistent souvent à combiner les différentes fonctions objectifs en une fonction d'utilité suivant les préférences du décideur. Dans ce cas le décideur est supposé connaître a priori le poids de chaque objectif afin de les mélanger dans une fonction unique. Cela revient à résoudre un problème simple objectif. Cependant dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, soit par manque d'expérience ou d'informations, soit parce que les différents objectifs sont non commensurables[23].

Les méthodes a posteriori (recherche → décideur)

Le décideur prend sa décision d'après un ensemble de solutions calculées par un solveur. Dans ce cas la qualité de la décision dépend du choix de la méthode de résolution. Car celle-ci va devoir donner un ensemble de résultats le plus représentatif de l'espace des objectifs efficaces.

Les méthodes progressives ou interactives (décideur ↔ recherche)

Dans ces méthodes, les processus de décision et d'optimisation sont alternés. Par moment, le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi interactivement le compromis entre ses

[22] Syn. Décideur.
[23] Des objectifs sont non commensurables s'ils sont exprimés dans des unités différentes.

préférences et les résultats. Ces méthodes exigent un connaissance approfondie, de la part du décideur, des outils utilisés. [Vincke 1988] présente plusieurs méthodes progressives utilisées en recherche opérationnelle. Il trouve que la relation de dominance[24] est trop pauvre pour être utile et les fonctions d'utilité multiattribut[25] trop riches pour être fiables. Il tente d'enrichir la relation de dominance par des éléments peu discutables : des préférences solidement établies.

2.4.2 Concepteur

Ce classement adopte un point de vue plus théorique articulé autour des notions d'agrégation et d'optimum de Pareto. Ces notions sont développées dans les sections suivantes car nous adoptons cette classification pour présenter les différentes méthodes.

Les méthodes agrégées

Ces méthodes transforment un problème multiobjectif en un problème simple objectif.

Les méthodes fondées sur Pareto

Ces méthodes sont fondées sur la notion de dominance au sens de Pareto[26] qui privilégie une recherche satisfaisant au mieux tous les objectifs.

Les méthodes non agrégées et non Pareto

Certaines méthodes n'utilisent aucun des deux concepts précédents. Alors que l'agrégation ou l'utilisation de la dominance de Pareto traitent les objectifs simultanément, en général, les méthodes dites non agrégées et non Pareto possèdent un processus de recherche qui traite séparément les objectifs.

2.5 Les méthodes agrégées

2.5.1 Théorie

2.5.1.1 Axiome fondamental

L'ensemble de ces méthodes repose sur l'axiome suivant : tout décideur essaye inconsciemment de maximiser une fonction d'utilité U.

$$U = U(f_1, f_2, \ldots, f_k) \qquad 8)$$

[24] La relation de dominance est essentiellement utilisée dans les méthodes a posteriori. Une définition est donnée au paragraphe 2.7.1.2.
[25] Terme utilisé pour identifier les méthodes d'agrégation.
[26] Cette notion est définie au paragraphe 2.7.1.

2.5.1.2 Les modèles additif et multiplicatif

Les modèles les plus couramment utilisés sont le modèle additif

$$U=\sum_{i=1}^{k} U_i(f_i) \qquad 9)$$

U_i fonction de mise à l'échelle du i^{ème} critère

et le modèle multiplicatif

$$\prod_{i=1}^{k} U_i(f_k) \qquad 10)$$

2.5.1.3 Difficultés issues de ces modèles

L'utilisation de ces modèles impose que les objectifs soient commensurables[27]. Il est donc très difficile d'utiliser ces techniques lorsque l'ensemble des critères est composé à la fois de critères qualitatifs et quantitatifs.

De plus, tout sous-ensemble d'objectifs doit être préférentiellement indépendant dans F.

Définition de l'indépendance préférentielle

Soit F l'ensemble des objectifs, J un sous-ensemble de f et $\bar{J}$ le complémentaire de J. J est préférentiellement indépendant dans F, si les préférences entre les actions qui ne diffèrent que par leurs valeurs sur les critères de J ne dépendent pas des valeurs des critères de $\bar{J}$.

Supposons quatre restaurants notés pour la préparation de trois plats.

Restaurant	Cuisses de grenouille	Steak tartare	Palourdes
a	18	15	19
b	15	18	19
c	18	15	11
d	15	18	11

Tableau 1 : Exemple de dépendance préférentielle, extrait de [Marichal 1999a]

En observant rapidement le tableau, nous établissons que $a \succ c$ et $b \succ d$. Mais en analysant les résultats, cette déduction est loin d'être évidente. Lorsqu'un restaurant est renommé pour la préparation des palourdes, nous préférons également qu'il soit bien noté pour la préparation des cuisses de grenouilles ($a \succ b$). Par contre, si un restaurant est mal noté pour la préparation des palourdes, nous privilégions comme second critère le restaurant bien noté pour la préparation du steak tartare ($d \succ c$). Nous constatons que les deux premiers critères (notes pour les cuisses de

[27] Exprimés dans la même unité.

grenouille et le steak tartare) ne sont pas préférentiellement indépendants du dernier (note pour les palourdes), car en fonction de la valeur de celui-ci les préférences sont inversées.

2.5.2 Les techniques

2.5.2.1 La moyenne pondérée

Cette méthode consiste à additionner tous les objectifs en affectant à chacun d'eux un coefficient de poids. Ce coefficient représente l'importance relative que le décideur attribue à l'objectif. Cela modifie un problème multiobjectif en un problème simple objectif de la forme :

$$\min \sum_{i=1}^{k} w_i f_i(x) \text{ avec } w_i \geq 0 \qquad 11)$$

w_i représente le poids affecté au critère i et $\sum_{i=1}^{k} w_i = 1$

Critique

Cette méthode est simple à mettre en œuvre et elle est d'une grande efficacité.

Mais les difficultés essentielles de cette approche sont :

1) Comment le décideur détermine-t-il les poids de chaque critère ?
2) Comment exprimer l'interaction entre les différents critères ?

Une solution au 1) est d'utiliser une combinaison linéaire des objectifs et de faire varier les poids de façon à constater l'influence de tel ou tel objectif sur le résultat. Cette approche est facile à implémenter mais tous les résultats obtenus appartiennent à des zones convexes de l'espace des objectifs réalisables (Figure 7). Les solutions potentielles situées dans des portions concaves sont oubliées.

Le deuxième problème est plus délicat à résoudre car il existe plusieurs types d'interaction entre deux critères i et j qui sont difficiles à exprimer à l'aide d'une somme pondérée :

- La corrélation positive ou négative. Le décideur souhaite que la contribution de j soit plus grande quand i n'est pas là (est là) car i et j sont corrélés.
- L'interchangeabilité. Le décideur souhaite que la satisfaction d'un seul critère produise presque le même effet que la satisfaction des deux critères.
- La complémentarité. Le décideur souhaite que la satisfaction d'un seul critère produise très peu d'effet par rapport à la satisfaction des deux critères.
- La dépendance préférentielle.

Dans [Marichal 1999a], l'auteur propose une axiomatique basée sur l'intégrale de Choquet [Choquet 1953] qui permet de prendre en compte les différentes interactions ci-dessus. Il a également proposé une méthode permettant de déterminer l'ensemble des poids d'interaction entre critères, et étendu ses travaux [Marichal 1999b, Marichal 2000] à l'agrégation de critères qualitatifs par utilisation de l'intégrale de Sugeno [Sugeno 1974].

L'utilisation de la somme pondérée se heurte également à deux problèmes : la sensibilité à un changement d'échelle et la compensation entre critères.

La sensibilité à un changement d'échelle. Cette transformation pourtant courante dans les modèles mathématiques peut s'avérer catastrophique lorsqu'elle est utilisée dans une somme pondérée car le changement d'échelle peut modifier l'ordre des différentes décisions.

Exemple tirée de [Maystre 1994]:

Considérons trois actions *a*, *b* et *c* évaluées en fonction d'un critère "coût" de poids 4/5 et d'un critère "quantité de déchets" de poids 1/5. Sachant que les déchets sont exprimés en tonne dans la colonne de gauche et en kilogramme dans celle de droite, on constate que ce changement d'échelle provoque une inversion des rangs des actions.

	Coût	**Déchets**		**Somme**		**Rang**	
a	100 000	5000	5 000 000	81 000	1 080 000	3	1
b	80 000	10000	10 000 000	66 000	2 064 000	2	2
c	40 000	20000	20 000 000	36 000	4 032 000	1	3

Tableau 2 : Détails des critères "coût" et "quantité de déchets"

La compensation entre les différents critères. Une valeur basse d'un critère peut être compensée par les valeurs des autres critères. Cela peut entraîner le décideur à faire le mauvais choix.

Exemple :

Considérons une commission de sélection chargée de choisir entre deux candidats. Le premier candidat a un total final de 77 points et le second de 76. La commission décide de prendre le premier. Est-ce que la décision aurait été la même si la commission avait eu connaissance des notes des candidats (le coefficient des épreuves est indiqué entre parenthèses).

Epreuve	**A(3)**	**B(2)**	**C(1)**	**Total**
Candidat 1	14	14	7	77
Candidat 2	13	13	11	76

Tableau 3 : Tableau des notes de chaque élève

Pour éviter ce genre de situation, la somme pondérée des résultats est souvent renforcée par des contraintes de seuil.

2.5.2.2 Goal programming

Cette méthode est également appelée **target vector optimisation** [Coello 1996, Van Veldhuizen 1999]. Le décideur fixe un but T_i à atteindre pour chaque objectif f_i [Charnes 1961]. Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre :

$$\min \sum_{i=1}^{k} \left| f_i(x) - T_i \right| \text{ avec } x \in F \qquad 12)$$

T_i représente la valeur à atteindre pour le i^{ème} objectif.

Différentes variantes et applications de cette technique ont été proposées [Ignizio 1981, Van Veldhuizen 1999].

Critique

Nous pouvons reprendre la critique faite pour la somme pondérée. La méthode est facile à mettre en œuvre mais la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode.

Cette méthode a l'avantage de fournir un résultat même si un mauvais choix initial a conduit le décideur a donné un ou plusieurs but(s) T_i non réalisable(s).

2.5.2.3 Le min-max

Cette méthode est assez proche de la précédente. Elle minimise le maximum de l'écart relatif entre un objectif et son but associé par le décideur.

$$\min \max_i \left(\frac{f_i(x) - T_i}{T_i} \right) \text{ avec } i = 1, \ldots, k \qquad 13)$$

T_i le but à atteindre pour le i^{ème} objectif.

w_i poids associé à un objectif.

Dans [Coello 1995], l'auteur présente précisément plusieurs variantes de la méthode min-max ainsi que diverses applications de celles-ci.

2.5.2.4 Goal attainment

Dans cette approche le décideur spécifie l'ensemble des buts T_i qu'il souhaite atteindre et les poids associés w_i. La solution optimale est trouvée en résolvant le problème suivant :

$$\text{minimiser } \alpha \text{ tel que} \qquad 14)$$

$$T_i + \alpha.w_i \geq f_i(x)$$

$$\text{avec } \sum_{i=0}^{k} w_i = 1$$

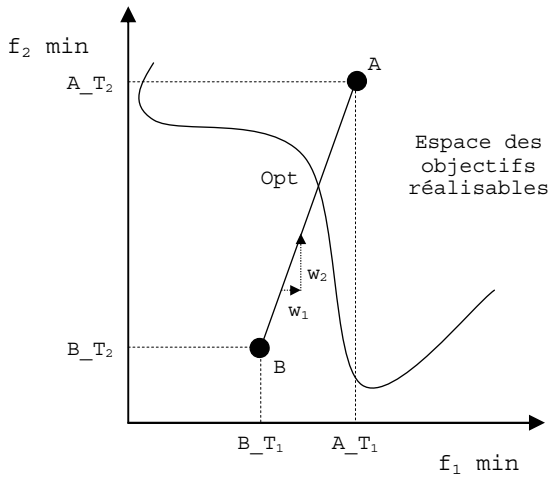


Figure 8 : Méthode goal attainment

Les objectifs T_i représentent le point de départ de la recherche dans l'espace et les poids w_i indiquent la direction de recherche dans l'espace. Dans la figure ci-dessus, *A* et *B* représentent deux buts à atteindre auxquels est associé le même couple de poids pour les deux fonctions objectifs. On remarque que si α est négatif, cela indique que le but peut être atteint alors que si α est positif le but ne pourra pas être atteint.

Critique

Contrairement à la somme pondérée, cette méthode peut générer des solutions sur la frontière de Pareto[28] en faisant varier les valeurs des w_i, même dans le cas d'une surface concave [Chen and Liu 1994].

[28] Cette notion est introduite au paragraphe 2.7.1.3 page 47.

2.5.2.5 La méthode ε-contrainte

Cette méthode est basée sur la minimisation d'un objectif f_i en considérant que les autres objectifs f_j avec $j \neq i$ doivent être inférieurs à une valeur ε_j. En général, l'objectif choisi est celui que le décideur souhaite optimiser en priorité.

$$\text{minimiser } f_i(x) \text{ avec} \qquad 15)$$

$$f_j(x) \leq \varepsilon_j, \forall\, j \neq i$$

De cette manière, un problème simple objectif sous contraintes peut être résolu. Le décideur peut ensuite réitérer ce processus sur un objectif différent jusqu'à ce qu'il trouve une solution satisfaisante. Cette méthode a été testée avec un algorithme génétique dans [Ritzel 1994] avec différentes valeurs de ε_j pour générer différentes valeurs Pareto-optimales.

Critique

La connaissance a priori des intervalles appropriés pour les valeurs de ε_j est exigée pour tous les objectifs.

2.6 Les méthodes non agrégées, non Pareto

2.6.1 Théorie

En général, les méthodes dites non agrégées et non Pareto possèdent un processus de recherche qui traite séparément les objectifs.

2.6.2 Les techniques

2.6.2.1 Vector Evaluated Genetic Algorithm (VEGA)

En 1985 Schaffer propose une extension d'un algorithme génétique simple pour la résolution d'un problème multiobjectif [Schaffer 1985]. Cette méthode est appelée Vector Evaluated Genetic Algorithm. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection. L'idée est simple. Si nous avons k objectifs et une population de n individus, une sélection de n/k individus est effectuée pour chaque objectif. Ainsi k sous-populations vont être créées, chacune d'entre elles contenant les n/k meilleurs individus pour un objectif particulier. Les k sous-populations sont ensuite mélangées afin d'obtenir une nouvelle population de taille n. Le processus se termine par l'application des opérateurs génétiques de modification (croisement et mutation).

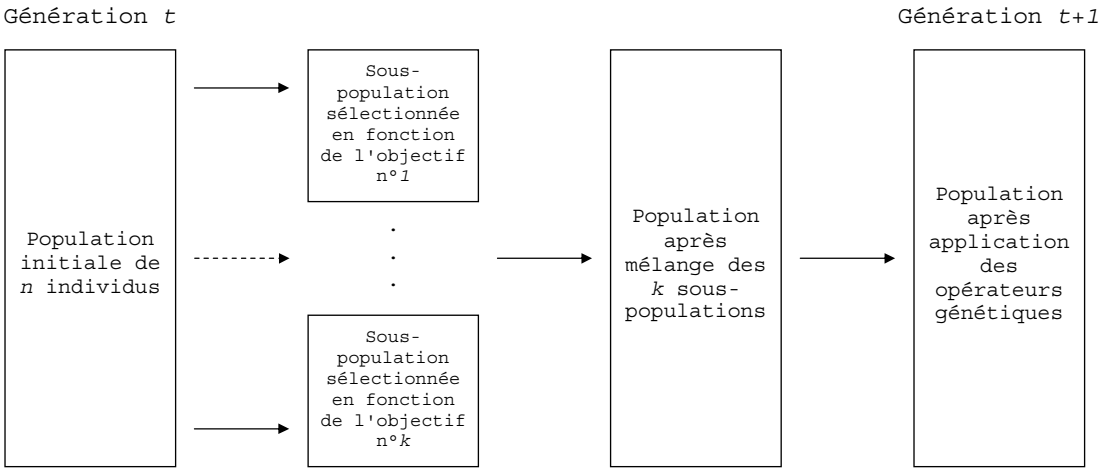


Figure 9 : Schéma de fonctionnement de VEGA

Discussion

La méthode VEGA a tendance à créer des sous-populations dont les meilleurs individus sont spécialisés pour un objectif particulier. L'évolution de la population favorise l'apparition des espèces. En effet, comme la méthode de sélection ne tient compte que d'un seul objectif, elle privilégie les individus qui obtiennent une bonne performance pour cet objectif. Dès lors ces individus ne seront sélectionnés que lorsqu'on effectuera la sélection sur cet objectif. Les individus que Schaffer appelle les individus "milieu", parce qu'ayant une performance générale acceptable mais ne possédant aucun critère fort, vont être éliminés car ils ne seront sélectionnés dans aucune sous-population. Cette disparition entraîne la spécialisation des individus pour chaque objectif. Ce résultat est contraire au but initial de la méthode qui était de trouver un compromis entre les différents critères.

Schaffer propose deux heuristiques pour améliorer sa méthode :

- La première est un croisement restreint qui ajoute une préférence pour sélectionner les parents non dominés. Cette méthode a tendance à éviter la disparition des individus "milieu" mais elle a tendance également à accentuer la convergence.
- La seconde encourage le croisement entre individus spécialisés sur des objectifs différents. Mais les effets sont identiques à la première heuristique.

Critique

Malgré ces imperfections, cette méthode est très souvent utilisée car facilement implémentable dans un algorithme génétique classique. L'utilisateur peut associer à VEGA n'importe quel mode de sélection (tournoi, roulette, rang). Mais comme le tournoi est une technique de sélection plus élitiste que les deux autres méthodes, son utilisation accentue le phénomène de spécialisation.

De nombreuses variations autour de cette technique ont été effectuées :

- mélange de VEGA avec dominance de Pareto [Tanaki 1995],
- paramètre pour contrôler le taux de sélection [Ritzel 1994],
- application à un problème contraint [Surry 1995],
- utilisation d'un vecteur contenant les probabilités d'utiliser un certain objectif lors de la sélection [Kurwase 1984].

2.6.2.2 Utilisation des genres

En 1992 Allenson propose une méthode qui utilise la notion de genre[29] et d'attracteur sexuel pour traiter un problème à deux objectifs [Allenson 1992]. Son exemple d'application consiste à minimiser la longueur d'un pipeline tout en réduisant l'impact écologique de sa construction. En affectant un objectif à chaque genre, l'auteur espère minimiser les deux objectifs simultanément car un genre sera toujours jugé d'après l'objectif qui lui a été associé.

Algorithme

Allenson utilise un algorithme génétique classique dans lequel un nombre égal d'individus des deux genres sera maintenu. La population est initialisée avec autant de males que de femelles, puis à chaque génération, les enfants remplacent les plus mauvais individus du même genre.

Allenson apporte une petite différence au niveau de la sélection. Il introduit le principe d'attracteur pour éviter que des individus trop éloignés au niveau de leur note puissent être croisés et pour permettre à des individus faibles de s'accoupler. Il espère ainsi maintenir plus longtemps la diversité dans la population et prévenir une convergence prématurée. En s'inspirant des stratégies d'évolution ($\mu+\lambda$), l'attracteur donne la capacité à chaque individu de s'accoupler avec un individu meilleur que lui. Dans son exemple, Allenson définit un attracteur comme un intervalle du type 10%-20% pour chaque individu et il utilise le rang de l'individu comme le point d'attraction génétique.

La création des enfants s'effectue par croisement mais leur genre est choisi aléatoirement et leur attracteur est créé en fonction de plusieurs heuristiques différentes (aléatoire, clonage ou croisement).

En 1996 Lis et Eiben ont également réalisé un algorithme basé sur l'utilisation des genres mais dans ce cas l'algorithme n'est pas limité à deux genres [Lis and Eiben 1996]. Il peut y avoir autant de genres que d'objectifs du problème. Ils ont également modifié le principe de croisement. Pour générer un enfant, un parent de chaque genre est sélectionné. Ensuite un croisement multipoint est effectué et le parent ayant participé le plus, en nombre de gènes, à l'élaboration de l'enfant transmet

son genre. En cas d'égalité le choix s'effectue aléatoirement entre les parents égaux. L'opérateur de mutation effectue un simple changement de genre.

Critique

L'utilisation de genre est un bon moyen de maintenir la diversité dans la population pour chaque objectif. De plus, le fait qu'un individu ne transmette pas systématiquement son genre va éviter la création d'espèces comme dans la méthode VEGA.

Par contre, l'augmentation du nombre d'objectifs accroît le temps de calcul car le croisement multipoint devient de plus en plus coûteux, alors que VEGA est très peu sensible à cela.

Le principe d'attracteur introduit par Allenson est une nouvelle façon d'effectuer des croisements restreints basée sur le phénotype des individus.

2.6.2.3 La méthode lexicographique

Fourman propose une méthode dans laquelle les objectifs sont préalablement rangés par ordre d'importance par le décideur [Fourman 1985]. Ensuite, l'optimum est obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite.

Expression mathématique du problème

Soient les fonctions objectifs f_i avec $i = 1, \ldots, k$, supposons un ordre tel que $f_1 \succ f_2 \succ \ldots \succ f_k$. Il faut :

$$\text{minimiser } f_1(x)$$

$$\text{avec } g_j(x) \text{ satisfait } \forall\, j = 1, \ldots, m$$

Soit x_1^*, la meilleure solution trouvée avec $f_1^* = f_1(x_1^*)$. f_1^* devient alors une nouvelle contrainte. L'expression du nouveau problème est donc :

$$\text{minimiser } f_2(x)$$

$$\text{avec } g_j(x) \text{ satisfait } \forall\, j = 1, \ldots, m$$

$$\text{et } f_1(x) = f_1^*$$

Soit x_2^* la solution de ce problème. Le $i^{ème}$ problème sera le suivant :

$$\text{minimiser } f_i(x) \qquad 16)$$

$$\text{avec } g_j(x) \text{ satisfait } \forall\, j = 1, \ldots, m$$

$$\text{et } f_1(x) = f_1^*, f_2(x) = f_2^*, \ldots, f_{(i-1)}(x) = f_{(i-1)}^*$$

[29] Genre : masculin ou féminin.

La procédure est répétée jusqu'à ce que tous les objectifs soient traités. La solution obtenue à l'étape *k* sera la solution du problème.

Fourman a proposé une autre version de cet algorithme qui choisit aléatoirement la fonction objectif devant être prise en compte. Il en déduit que cela marche aussi bien. Cette façon de procéder équivaut à une somme pondérée dans laquelle un poids correspond à la probabilité que la fonction objectif associée soit sélectionnée.

Critique

Le risque essentiel de cette méthode est la grande importance attribuée aux objectifs classés en premier. La meilleure solution f_1^* trouvée pour l'objectif le plus important va faire converger l'algorithme vers une zone restreinte de l'espace d'état et enfermer les points dans une niche[30].

2.6.2.4 A Non Generational Genetic Algorithm

Valenzuela et Uresti proposent une sélection des individus non générationnelle dans laquelle la fitness est calculée de façon incrémentale [Valenzuela and Uresti 1997]. Ils appliquent leur méthode sur une conception de matériel électronique dans laquelle ils doivent maximiser la performance du matériel, minimiser le temps moyen entre deux erreurs et minimiser le coût de revient. Ces objectifs étant non commensurables, ils ont recherché une méthode permettant d'outrepasser ce problème. L'idée d'utiliser un algorithme non générationnel provient des systèmes de classifeurs[31] [Goldberg 1989]. Ils pensent qu'un problème multiobjectif a des caractéristiques similaires à celles des systèmes de classifieurs, et donc qu'une approche non générationnelle peut être plus performante qu'un algorithme générationnel.

Un classifieur possède une caractéristique, appelée force, qui représente son adaptation à son environnement. Cette force évolue dans le temps de façon incrémentale en fonction des récompenses obtenues lors de réponses positives à des stimuli. [Valenzuela and Uresti 1997] proposent de transposer ce comportement à des individus qui recherchent la frontière de Pareto d'un problème multiobjectif.

Leur algorithme est décomposé en deux parties. Une première partie dans laquelle la fitness des individus est mise à jour et une seconde partie qui crée de nouveaux individus après sélection, croisement et mutation.

L'évaluation d'un individu *i* est basée sur deux mesures, δ_i et w_i, calculées incrémentalement. Ces mesures sont mises à jour chaque fois que l'on trouve un individu *j* qui domine *i*.

[30] Analogie avec les niches écologiques : ensemble d'individus situés dans un espace restreint.
[31] Syn. LCS : Learning Classifier System.

δ_i évalue la domination de l'individu au cours du temps :

$$\delta_i(t+1) = \delta_i(t) - k_\delta \delta_i(t) + D(t) \quad 17)$$

$D(t)$ est égal à *1* si l'individu est dominé, *0* sinon.

w_i évalue le voisinage de l'individu par une fonction de sharing :

$$w_i(t+1) = w_i(t) - k_w.w_i(t) + \text{sh}(d) \quad 18)$$

sh(d), calcul de la distance entre les deux individus

La fonction de fitness de chaque individu est calculée en fonction de δ_i et w_i :

$$fitness = c_d.\delta_i + c_w.w_i \quad 19)$$

Les auteurs réduisent ainsi un problème multiobjectif à un problème à deux objectifs :

1) minimiser la dominance,

2) minimiser l'importance de la niche.

c_d et c_w sont des constantes choisies arbitrairement qui expriment le compromis entre ces 2 objectifs.

Si l'on compare les équations ci-dessus à l'équation d'évolution de la force d'un classifieur, $D(t)$ et sh(d) représentent la récompense du classifieur et les paramètres k_δ et k_w représentent le coût d'activation d'un classifieur. Dans leurs expérimentations, les auteurs affectent sans aucune justification, les valeurs *0* et $1/n_1$ respectivement à k_δ et k_w. De manière générale, ces deux paramètres peuvent être utilisés afin d'atténuer l'influence des valeurs passées sur les valeurs présentes.

Algorithme

/*----- Initialisation -----*/
Génération aléatoire de la population initiale *pop*
Pour chaque individu $i \in pop$
  $\delta_i \leftarrow 0$
  $w_i \leftarrow 0$
  $N \leftarrow n_1$ /*----- Sélection aléatoire de n_1 Individus -----*/
  Pour chaque $j \in N$
    Si j domine i **alors** $\delta_i \leftarrow \delta_i + 1$
    Si i domine j **alors** $\delta_j \leftarrow \delta_j + 1$
    $w_i \leftarrow \delta_i + \text{sh}(d_{ij})$
    $w_j \leftarrow \delta_j + \text{sh}(d_{ij})$

Fin Pour

Fin Pour

/*----- Boucle principale -----*/

Répéter

Pour i **de** 1 **à** n_2

/*----- Evaluation -----*/

Deux individus i et j sont choisis aléatoirement

Si j domine i **alors** $D_i \leftarrow 1$ **sinon** $D_i \leftarrow 0$

Si i domine j **alors** $D_j \leftarrow 1$ **sinon** $D_j \leftarrow 0$

$\delta_i \leftarrow \delta_i - k_\delta.\delta_i + D_i$

$\delta_j \leftarrow \delta_j - k_\delta.\delta_j + D_j$

$w_i \leftarrow w_i - k_w.w_i + \mathrm{sh}(d_{ij})$

$w_j \leftarrow w_j - k_w.w_j + \mathrm{sh}(d_{ij})$

$f_i \leftarrow c_\delta.\delta_i + c_w.w_i$

$f_j \leftarrow c_\delta.\delta_j + c_w.w_j$

Fin Pour

/*----- Sélection, croisement et mutation -----*/

$f_{max} \leftarrow \max(f_i)$

Sélection de deux parents proportionnellement à $(f_{max} - f_i)$

Création d'un nouvel individu par croisement et mutation

Supprime un individu i tel que $f_i = f_{max}$

Soit i le nouvel individu

$N \leftarrow n_1$ /*----- Sélection aléatoire de n_1 Individus -----*/

Pour chaque $j \in N$

Si j domine i **alors** $\delta_i \leftarrow \delta_i + 1$

Si i domine j **alors** $\delta_j \leftarrow \delta_j + 1$

$w_i \leftarrow \delta_i + \mathrm{sh}(d_{ij})$

$w_j \leftarrow \delta_j + \mathrm{sh}(d_{ij})$

Fin Pour

Jusqu'à un critère de terminaison

Critique

Cette méthode non générationnelle obtient de meilleurs résultats que NPGA[32] dans trois optimisations à deux objectifs en terme de nombre de points sur la surface de Pareto et en nombre total d'évaluations.

Même si cette technique obtient de bons résultats, elle apparaît moins flexible que les précédentes. Il apparaît difficile pour un décideur de pouvoir exprimer ses préférences sur un ou plusieurs objectif(s). De plus, l'introduction de nombreux paramètres (c_d, c_w, k_δ, k_w, n_1, n_2) supplémentaires, en plus des paramètres liés à l'algorithme génétique, rend difficile le réglage de l'algorithme.

2.6.2.5 Une méthode élitiste

En 1996 Ishibuchi et Murata proposent un algorithme basé sur une sélection de type min-max dans lequel les solutions non dominées[33] trouvées à chaque génération sont stockées dans une population externe [Ishibuchi and Murata 1996]. Les auteurs utilisent également une méthode de recherche locale pour générer de meilleurs individus.

Algorithme

a) Initialisation aléatoire de la population courante appelée *CURRENT* de taille *M*.

b) Calcul des notes des individus et stockage des individus non dominés dans une population externe appelée *NOND* de taille *N*.

c) Sélection de (*M*-*N*) paires de parents dans *CURRENT*.

d) Croisement et mutation.

e) Sélection d'individus appartenant à *NOND* et ajout à *CURRENT*.

f) Application d'une stratégie de recherche locale sur tous les individus de la population.

g) Retour à b) si le critère d'arrêt n'est pas trouvé.

Critique

L'utilisation d'une population externe pour stocker les individus non dominés et d'une recherche locale apporte à cette méthode une capacité élitiste très importante.

Nous allons voir dans la section suivante que l'introduction de ce mécanisme de stockage associé aux stratégies de mise à jour de cette population externe et de réinjection des individus dans la population courante va inspirer beaucoup de chercheurs.

[32] Méthode présentée au paragraphe 2.7.2.3 page 52.

[33] La notion de dominance est introduite dans le paragraphe suivant. Bien que Ishibuchi et Murata utilise cette notion, leur méthode n'est pas classée dans les méthodes de Pareto car la notion de dominance n'est pas utilisée dans le processus de sélection.

2.7 Les méthodes Pareto

L'idée d'utiliser la dominance au sens de Pareto a été proposée par Goldberg [Goldberg 1989] pour résoudre les problèmes proposés par Schaffer [Schaffer 1985]. Il suggère d'utiliser le concept d'optimalité de Pareto pour respecter l'intégralité de chaque critère car il refuse de comparer a priori les valeurs de différents critères. L'utilisation d'une sélection basée sur la notion de dominance de Pareto va faire converger la population vers un ensemble de solutions efficaces. Ce concept ne permet pas de choisir une alternative plutôt qu'une autre mais il apporte une aide précieuse au décideur.

Dans les paragraphes suivants, nous définissons tout d'abord la notion de dominance au sens de Pareto, la frontière de Pareto et la notion de domination contrainte. Ensuite, nous présentons les techniques évolutionnaires utilisant cette notion.

2.7.1 Théorie

2.7.1.1 Optimum de Pareto

Au XIX^{ième} siècle, Vilfredo Pareto, un mathématicien italien, formule le concept suivant [Pareto 1896] : dans un problème multiobjectif, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères.

Cette équilibre a été appelé optimum de Pareto. Un point x est dit **Pareto-optimal** s'il n'est dominé par aucun autre point appartenant à E. Ces points sont également appelés solutions *non inférieures* ou *non dominées*.

2.7.1.2 La notion de dominance

Un point $x \in E$ **domine** $x' \in E$ si

$$\forall\, i, f_i(x) \leq f_i(x') \text{ avec} \qquad 20)$$

$$\text{au moins un } i \text{ tel que } f_i(x) < f_i(x')$$

Dans l'exemple ci-dessous, les points 1,3 et 5 ne sont dominés par aucun autre. Alors que le point 2 est dominé par le point 1, et que le point 4 est dominé par les points 3 et 5.

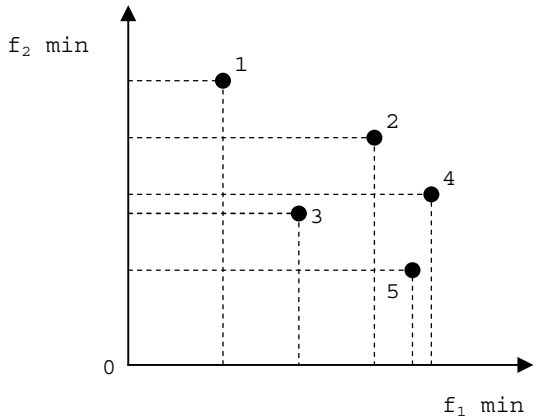


Figure 10 : Exemple de dominance

Un point $x \in E$ est dit **faiblement non dominé**, si il n'existe pas de point $x' \in E$ tel que :

$$f_i(x') < f_i(x), \forall\ i = 1, \ldots, k$$

Un point $x \in E$ est dit **fortement non dominé**, si il n'existe pas de point $x' \in E$ tel que :

$$f_i(x') \leq f_i(x), \forall\ i = 1, \ldots, k \text{ avec}$$

$$\text{au moins un } i \text{ tel que, } f_i(x') < f_i(x)$$

2.7.1.3 La frontière de Pareto

La frontière de Pareto est l'ensemble de tous les points Pareto-optimaux. Les figures ci-dessous présentent pour un problème à deux objectifs les quatre frontières de Pareto en fonction du désir de l'utilisateur de minimiser ou maximiser les objectifs.

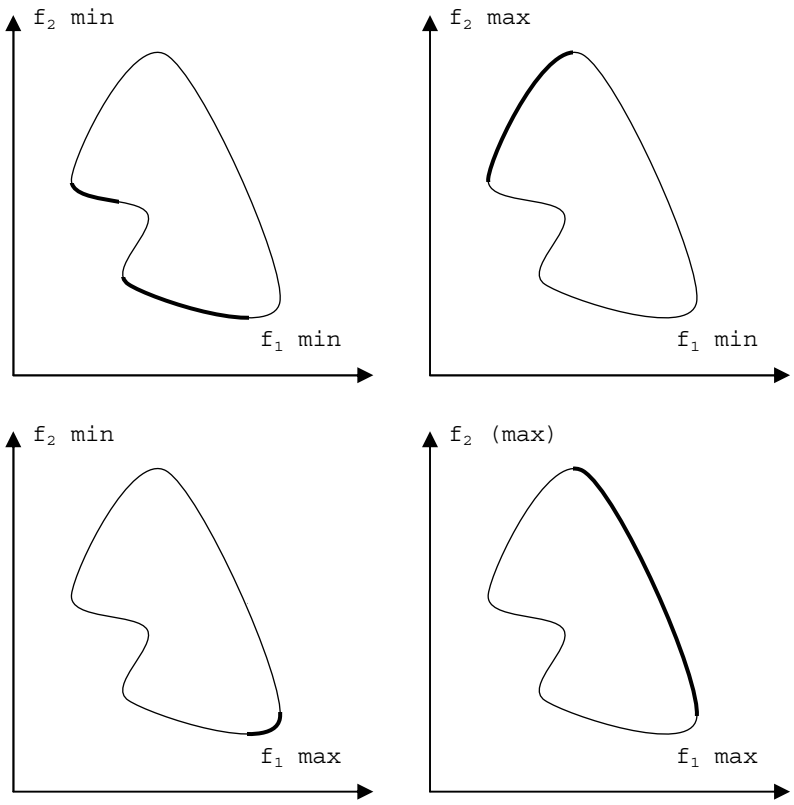


Figure 11 : Exemples de frontière de Pareto

Dans l'exemple de la Figure 10, la frontière de Pareto est composée des points 1,3 et 5.

2.7.1.4 La notion de domination-contrainte

La notion de domination-contrainte a été introduite pour enrichir la notion de dominance dans les problèmes multiobjectifs contraints. La plupart des méthodes de résolution de problèmes uniobjectifs utilisent une approche basée sur l'attribution de pénalités en fonction des contraintes violées. Mais cette approche a deux inconvénients :

- elle exige de fixer les règles de pénalité,
- elle ne fait pas la différence entre deux individus ayant la même note dont l'un viole des contraintes et l'autre pas.

Une première définition de la domination contrainte a été donnée par Jiménez et Verdegay [Jiménez and Verdegay 1998]. Ils utilisent un tournoi à deux individus *i* et *j*. Si *i* donne une solution réalisable et *j* donne une solution non réalisable alors *i* est choisi. Si les deux individus donnent des solutions réalisables alors ils sont comparés avec un sous ensemble d'individus choisis aléatoirement[34]. Si les individus donnent des solutions non réalisables alors ils sont comparés avec un ensemble d'individus du même type. Le meilleur des deux ou celui qui est situé dans la plus petite niche sera sélectionné.

[34] Cette technique de tournoi a été introduite par Horn et Napfliotis (voir le paragraphe 2.7.2.3).

Une autre définition a été proposée dans [Deb and al 2000]. Une solution *i* domine avec contrainte une solution *j* si un des cas suivants est réalisé :

- La solution *i* est réalisable et la solution *j* ne l'est pas.
- Les deux solutions *i* et *j* ne sont pas réalisables mais *i* a le plus petit total de violation de contrainte.
- Les deux solutions sont réalisables mais *i* domine *j*.

L'utilisation de cette notion exige que l'on puisse mesurer le niveau de violation de chaque contrainte du problème.

2.7.2 Les techniques non élitistes

2.7.2.1 Multiple Objective Genetic Algorithm (MOGA)

En 1993 Fonseca et Fleming ont proposé une méthode dans laquelle chaque individu de la population est rangé en fonction du nombre d'individus qui le dominent [Fonseca and Fleming 1993]. Ensuite, ils utilisent une fonction de notation permettant de prendre en compte le rang de l'individu et le nombre d'individus ayant même rang.

Soit un individu x_i à la génération t, dominé par $p_i(t)$ individus. Le rang de cet individu est :

$$\text{rang}(x_i, t) = 1 + p_i(t) \qquad 21)$$

Tous les individus non dominés sont de rang 1. Dans l'exemple (Figure 10), le tableau de dominance est le suivant :

Point	Dominé par	Rang
1	aucun	1
2	1	2
3	aucun	1
4	3 et 5	3
5	aucun	1

Tableau 4 : Tableau de dominance

[Fonseca and Fleming 1993] calculent la fitness de chaque individu de la façon suivante :

a) Calcul du rang de chaque individu.

b) Affectation de la fitness de chaque individu par application d'une fonction de changement d'échelle[35] sur la valeur de son rang. Cette fonction est en général linéaire. Suivant le problème, d'autres types de fonction pourront être envisagés afin d'augmenter ou de

[35] On parle aussi de fonction de scaling.

diminuer l'importance des meilleurs rangs ou d'atténuer la largeur de l'espace entre les individus de plus fort rang et de plus bas rang.

Discussion

L'utilisation de la sélection par rang a tendance à répartir la population autour d'un même optimum. Or cela n'est pas satisfaisant pour un décideur car cette méthode ne lui proposera qu'une seule solution. Pour éviter cette dérive, les auteurs utilisent une fonction de *sharing*[36]. Ils espèrent ainsi répartir la population sur l'ensemble de la frontière de Pareto.

Critique

Le sharing utilisé dans cette méthode agit sur l'espace des objectifs. Cela suppose que deux actions qui ont le même résultat dans l'espace des objectifs ne pourront pas être présentes dans la population.

Cette méthode obtient des solutions de bonne qualité et son implémentation est facile. Mais les performances sont dépendantes de la valeur du paramètre σ_{share} utilisé dans le sharing. Dans leur article Fonseca et Fleming expliquent comment choisir au plus juste la valeur de σ_{share}.

2.7.2.2 Non dominated Sorting Genetic Algorithm (NSGA)

Dans la méthode proposée par [Srivinas and Deb 1993], le calcul de la fitness s'effectue en séparant la population en plusieurs groupes en fonction du degré de domination au sens de Pareto de chaque individu.

Algorithme de la fonction de notation

a) Dans la population entière, on recherche les individus non dominés. Ces derniers constituent la première frontière de Pareto.

b) On leur attribue une valeur de fitness factice. Cette valeur est supposée donner une chance égale de reproduction à tous ces individus. Mais pour maintenir la diversité dans la population, il est nécessaire d'appliquer une fonction de sharing sur cette valeur.

c) Ensuite, ce premier groupe d'individus est supprimé de la population.

d) On recommence cette procédure pour déterminer la seconde frontière de Pareto. La valeur factice de fitness attribuée à ce second groupe est inférieure à la plus petite fitness après application de la fonction de sharing sur le premier groupe. Ce mécanisme est répété jusqu'à ce que l'on ait traité tous les individus de la population.

[36] Cette technique est décrite dans le paragraphe 3.5.2.2. On parle également d'heuristique de partage.

L'algorithme se déroule ensuite comme un algorithme génétique classique. Srivinas et Deb utilisent une sélection basée sur le reste stochastique. Mais leur méthode peut être utilisée avec d'autres heuristiques de sélections (tournoi, roulette pipée, etc.).

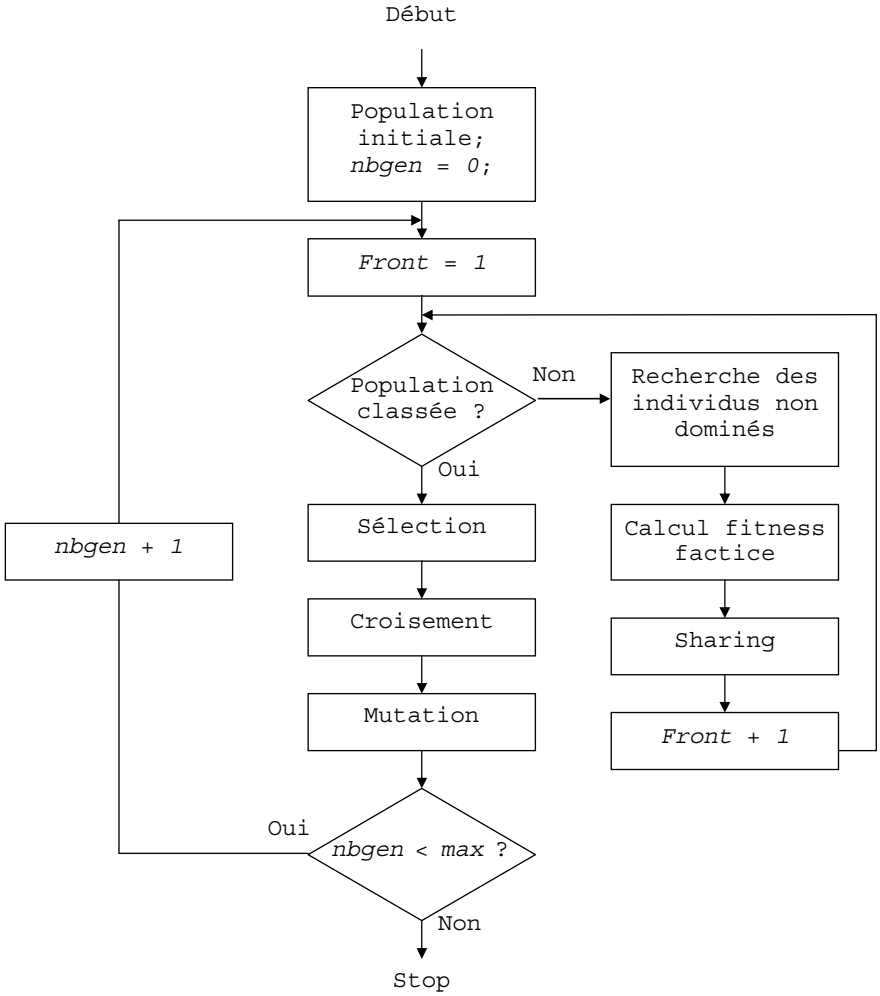


Figure 12 : Schéma de fonctionnement de NSGA

Critique

Cette méthode paraît moins efficace en temps de calcul que la méthode MOGA car le temps de calcul de la notation (tri de la population et sharing) est important. Mais l'utilisation d'un sharing sur l'espace d'état et le tri des solutions en différentes frontières semblent plus appropriés à maintenir une grande diversité de la population et à répartir plus efficacement les solutions sur la frontière de Pareto. De plus, cette méthode est applicable dans des problèmes avec un nombre quelconque d'objectifs.

Trois critiques ont été soulevées pour cette méthode :

- Sa complexité de calcul de $O(k.N^3)$ avec k le nombre d'objectifs et N la taille de la population, essentiellement due au processus de tri de la population et d'application de l'heuristique de partage.
- Son approche non élitiste[37]. Le tri de la population est une heuristique intéressante pour distribuer la population sur l'ensemble de la frontière de Pareto mais cette procédure ralentit le processus de convergence de l'algorithme. De plus, cet effet est accentué par l'utilisation de la méthode de sélection par reste stochastique.
- La nécessité de spécifier un paramètre de sharing.

2.7.2.3 Niched Pareto Genetic Algorithm (NPGA)

Cette méthode proposée par Horn et Napfliotis utilise un tournoi basé sur la notion de dominance de Pareto [Horn and Napfliotis 1993]. Elle compare deux individus pris au hasard avec une sous-population de taille t_{dom} également choisie au hasard. Si un seul de ces deux individus domine le sous-groupe, il est alors positionné dans la population suivante. Dans les autres cas une fonction de sharing est appliquée pour sélectionner l'individu.

Le paramètre t_{dom} permet d'exercer une pression variable sur la population et ainsi d'augmenter ou de diminuer la convergence de l'algorithme. Par exemple, si $t_{dom} = 2$ la convergence sera très lente. A travers leurs expérimentations [Horn and Napfliotis 1993] établissent le constat suivant :

- Si $t_{dom} \approx 1\%$ de N (nombre d'individus de la population), il y a trop de solutions dominées. Cette valeur n'apporte pas assez d'élitisme dans la sélection, d'où un grand nombre de solutions non dominées qui subsistent dans la population.
- Si $t_{dom} \approx 10\%$ de N, une bonne distribution des individus est obtenue. Cette valeur apporte un bon compromis entre élitisme et représentativité statistique de la population. Elle permet non seulement d'exercer une pression suffisante sur la sélection des individus, mais également d'éviter une convergence non désirée due à une sélection effectuée sur un échantillon non représentatif de la population.
- Si $t_{dom} >> 20\%$ de N, il y a une convergence prématurée, car la pression est trop importante lors de la sélection.

On retrouve des conclusions identiques, à l'échelle près, à l'action du paramètre t_{size}[38] sur l'élitisme de la sélection par tournoi.

[37] Comme nous le verrons dans les paragraphes suivants, les auteurs ont tendance à qualifier d'élitiste : une méthode qui utilise une population externe pour stocker les solutions Pareto-optimales trouvées jusqu'alors.
[38] Nombre d'individus participant au tournoi.

Algorithme de la fonction de sélection

sous_pop[] ← Sélection aléatoire de t_{dom} + *2*

candidat1 ← *sous_pop*[*1*]

candidat2 ← *sous_pop*[*1*]

cand_dominé1 ← *faux*

cand_dominé2 ← *faux*

Pour *i* **de** *3* **à** t_{dom} + *2* **faire**

cand_comp ← *sous_pop*[*i*]

Si (*cand_comp* domine *cand_dominé1*) **alors** *cand_dominé1* ← *vrai*

Si (*cand_comp* domine *cand_dominé2*) **alors** *cand_dominé2* ← *vrai*

Fin Pour

Si (*cand_dominé1* **et** non *cand_dominé2*)

alors retourne *cand_dominé2*

sinon si (non *cand_dominé1* **et** *cand_dominé2*)

alors retourne *cand_dominé1*

sinon appliquer l'heuristique de sharing

Critique

Outre les paramètres de sharing l'utilisation de cette méthode ajoute un paramètre supplémentaire à fixer par l'utilisateur, t_{dom}. Ce dernier permet de régler aisément la pression sur la population. Mais il faut faire attention au compromis entre représentativité et élitisme car la taille de l'échantillon sélectionné est plus influente dans ce cadre de problème que dans une utilisation de la sélection par tournoi classique.

Les solutions trouvées sont de bonne qualité et cette approche est plus rapide que les approches précédentes car le sharing n'est appliqué que sur une portion de la population.

2.7.3 Les techniques élitistes

Les approches que nous venons de voir sont dites non élitistes car :

1) Elles ne conservent pas les individus Pareto-optimaux trouvés au cours du temps.
2) Elles maintiennent difficilement la diversité sur la frontière de Pareto.
3) La convergence des solutions vers la frontière de Pareto est lente.

Pour résoudre les difficultés ci-dessus, de nouvelles techniques ont été appliquées.

1) Introduction d'une population externe ou archive permettant de stocker les individus Pareto-optimaux.

2) Utilisation de techniques de niching, clustering et grid-based pour répartir efficacement les solutions sur la frontière de Pareto.

3) Préférence pour les solutions non dominées.

Les paragraphes suivants présentent les différentes évolutions des méthodes élitistes.

2.7.3.1 Strength Pareto Evolutionary Algorithm (SPEA)

En 1998 Zitzler et Thiele proposent une nouvelle méthode d'optimisation multiobjectif qui possède les caractéristiques suivantes [Zitzler and Thiele 1998] :

- Utilisation du concept de Pareto pour comparer les solutions.
- Un ensemble de solutions Pareto-optimales est maintenu dans une population externe appelée **archive**.
- La fitness de chaque individu est calculée par rapport aux solutions stockées dans l'archive.
- Toutes les solutions de l'archive participent à la sélection.
- Une méthode de clustering[39] est utilisée pour réduire l'ensemble de Pareto sans supprimer ses caractéristiques.
- Une nouvelle méthode de niche, basée sur Pareto, est utilisée afin de préserver la diversité. L'avantage essentiel est qu'elle n'exige pas de réglage de paramètres de sharing.

Algorithme

Le passage d'une génération à une autre commence par la mise à jour de l'archive. Tous les individus non dominés sont copiés dans l'archive et les individus dominés déjà présents sont supprimés. Si le nombre d'individus dans l'archive excède un nombre donné, on applique une technique de clustering pour réduire l'archive. Ensuite la fitness de chaque individu est mise à jour avant d'effectuer la sélection en utilisant les deux ensembles. Pour terminer, on applique les opérateurs génétiques de modification.

[39] Cette technique est décrite en annexe page 169.

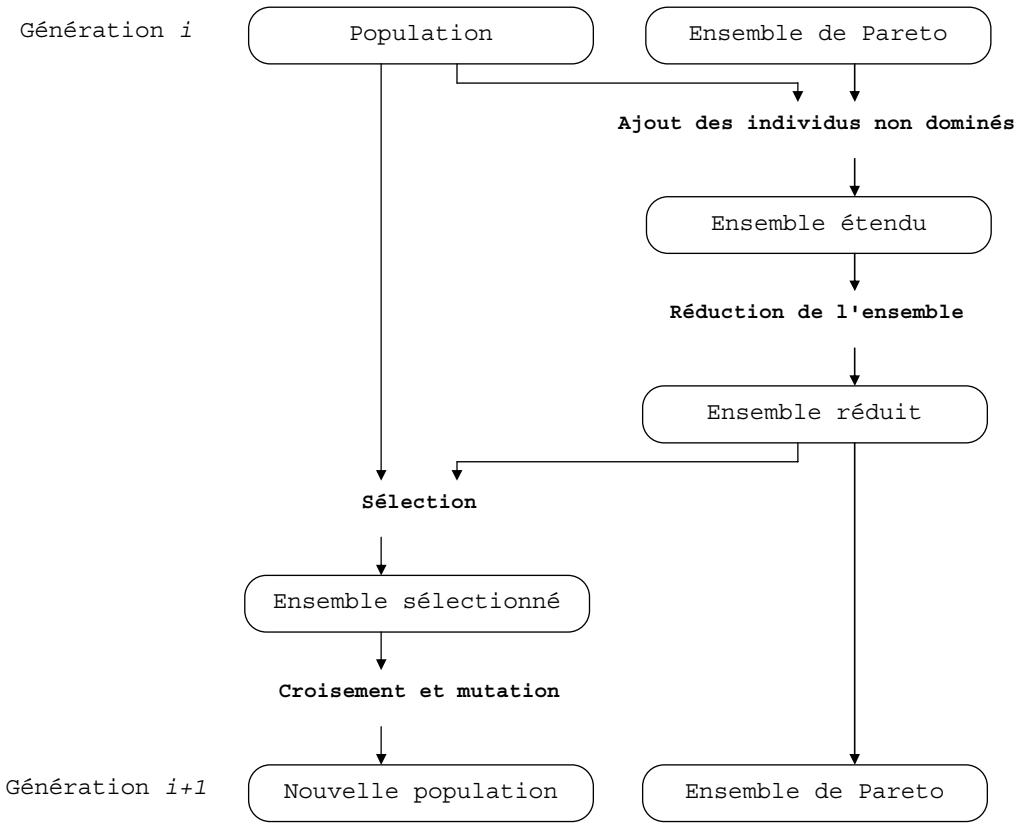


Figure 13 : Schéma de fonctionnement de SPEA

Calcul de la fitness des individus

Le calcul de la notation s'effectue en deux temps.

1) On attribue une valeur $s \in [0, 1]$, appelée force[40], à chaque individu de l'ensemble Pareto-optimal. Cette valeur est proportionnelle au nombre d'individus de la population qu'il domine. s est égale au nombre d'individus dominés, divisé par le nombre d'individus de la population + 1. La force s représente également la fitness des solutions Pareto-optimales.

2) La fitness f_i d'un individu de la population est égale à la somme des forces des individus Pareto-optimaux qui le dominent + 1.

[40] Ce terme est tiré des systèmes de classifieurs.

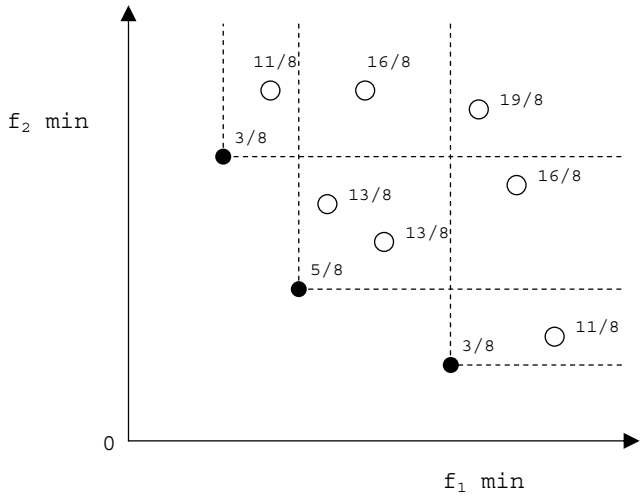


Figure 14 : Exemple de notation avec SPEA

Critique

Cette méthode distribue efficacement les solutions sur la frontière de Pareto. La technique de notation permet de bien échantillonner les individus dans l'espace (Figure ci-dessus). Le concept de force associé à la technique de clustering entraîne la formation de niches dont la note des individus dépend de leur position par rapport aux individus Pareto-optimaux.

Le point négatif est que la notation est dépendante de la taille de la population externe choisie par l'utilisateur.

2.7.3.2 Pareto Archived Evolution Strategy (PAES)

Cette méthode a été développée initialement comme une méthode de recherche locale dans un problème de routage d'information off-line. Les premiers travaux de Knowles et Corne ont montré que cette méthode simple objectif fournissait des résultats supérieurs aux méthodes de recherche basées sur une population [Knowles and Corne 1999]. Par conséquent, les auteurs ont adapté cette méthode aux problèmes multiobjectifs. Les particularités de cette méthode sont les suivantes :

- Elle n'est pas basée sur une population. Elle n'utilise qu'un seul individu à la fois pour la recherche des solutions.
- Elle utilise une population annexe de taille déterminée permettant de stocker les solutions temporairement Pareto-optimales.
- L'algorithme utilisé est très simple et inspiré d'une stratégie d'évolution (*1+1*) [Rechenberg 1973].

- Elle utilise une technique de *crowding*[41] basée sur un découpage en hypercubes de l'espace des objectifs.

Algorithme

L'algorithme de PAES se présente en trois parties [Knowles and Corne 2000a] :

1) Génération d'une solution candidate.
2) Fonction d'acceptation de la solution candidate.
3) Archivage des solutions non dominées.

La méthode de génération d'un nouveau candidat ressemble à la méthode de Hill-Climbing. A chaque itération un nouveau candidat est produit par mutation aléatoire.

a) Génération aléatoire d'une solution *c* et ajout de *c* à l'archive.
b) Production d'une solution *m* par mutation de *c* et évaluation de *m*.
c) **Si** (*c* domine *m*)
   alors on écarte *m*.
   sinon si (*m* domines *c*)
      alors on remplace *c* par *m* et ajout de *m* à l'archive.
      sinon si (*m* est dominé par un membre de l'archive)
         alors on écarte *m*.
         sinon on applique une fonction de test (*c*, *m*, archive) qui détermine la nouvelle solution courante.
d) On recommence en b)

Fonction de test (*c*, *m*, archive)

a) **Si** l'archive n'est pas pleine
   alors ajout de *m* à l'archive.
   sinon si (*m* est dans une région moins encombrée qu'une solution $x \in$ à l'archive)
      alors ajout de *m* et suppression d'un membre de la zone la plus encombrée de l'archive.
b) **Si** (*m* est dans une région moins encombrée que *c*)
   alors *m* est acceptée comme solution courante.
   sinon on conserve *c* comme solution courante.

[41] Cette technique est décrite dans le paragraphe 3.5.2.3.

Pour mesurer l'encombrement d'une zone, cette méthode utilise un crowding basé sur un découpage en hypercubes de l'espace des objectifs. Cette technique offre deux avantages par rapport aux méthodes de sharing classiques :

- le temps de calcul est moins important,
- la découpage étant adaptatif, cela ne nécessite pas de réglage de paramètre.

Une grille divise l'espace des phénotypes en hypercubes, chaque dimension est découpée en $k_{range}/2^l$ parties. k_{range} représente l'écart pour un objectif entre la plus grande valeur et la plus faible dans l'archive, et l est le paramètre de subdivision. Les auteurs utilisent une représentation récursive[42] de la grille afin d'accélérer la recherche ou la mise à jour de la position d'une solution dans l'espace. Mais cette représentation devra être totalement recalculée lorsque l'ajout d'un individu dans l'archive entraînera le changement d'un k_{range}.

Dans leur article, une généralisation à un algorithme $(\mu+\lambda)$-PAES est proposée. Les tests effectués par les auteurs montrent que les algorithmes $(1+\lambda)$-PAES et $(\mu+\lambda)$-PAES n'apportent pas un gain significatif par rapport à $(1+1)$-PAES.

Critique

Cette méthode est relativement simple à mettre en œuvre. De plus, n'étant pas basée sur un algorithme génétique, elle évite à l'utilisateur le réglage de tous les paramètres de celui-ci. Mais son efficacité va dépendre du choix d'un nouveau paramètre : l le paramètre de discrétisation de l'espace des objectifs.

La technique de crowding utilisée dans PAES permet une mise à jour de l'archive plus rapide lors des dépassements de capacité que celle de SPEA.

La complexité de cet algorithme est $O(a.k.N)$ avec a la taille de l'archive $\approx O(k.N^2)$.

2.7.3.3 Pareto Envelope based Selection Algorithm (PESA)

La méthode PESA a été également proposée par Knowles et Corne [Knowles and Corne 2000b]. Elle reprend approximativement le principe de crowding développé dans PAES et définit un paramètre appelé *squeeze_factor* qui représente la mesure d'encombrement d'une zone de l'espace. Alors que PAES est basée sur une stratégie d'évolution, PESA est une méthode basée sur les algorithmes génétiques. Elle définit deux paramètres concernant la taille des populations d'individus : P_I (taille de la population interne) et P_E (taille de la population externe ou archive).

Algorithme

a) Génération aléatoire et évaluation de la population P_I, et initialisation de P_E ($P_E = \varnothing$).

[42] Dans un problème à deux objectifs, la grille sera représentée sous forme de quadtree.

b) Transfert de tous les individus non dominés de P_I dans P_E.

c) **Si** le critère d'arrêt est réalisé

alors on retourne P_E comme ensemble de solutions.

sinon on supprime tous les individus de P_I et on recrée P_I de la façon suivante :

On sélectionne deux parents dans P_E avec la probabilité p_c, on produit un enfant par croisement puis on le mute. Avec la probabilité (1-p_c), on sélectionne un parent et on le mute pour produire un autre enfant.

d) On recommence au b)

Une solution courante de P_I peut entrer dans l'archive P_E si elle est non dominée dans P_I et si elle est non dominée dans P_E. Une fois, la solution insérée dans l'archive, on supprime tous les membres de l'archive qu'elle domine. Si l'ajout crée un dépassement de capacité de P_E alors le membre de l'archive ayant le paramètre *squeeze_factor* le plus élevé est supprimé.

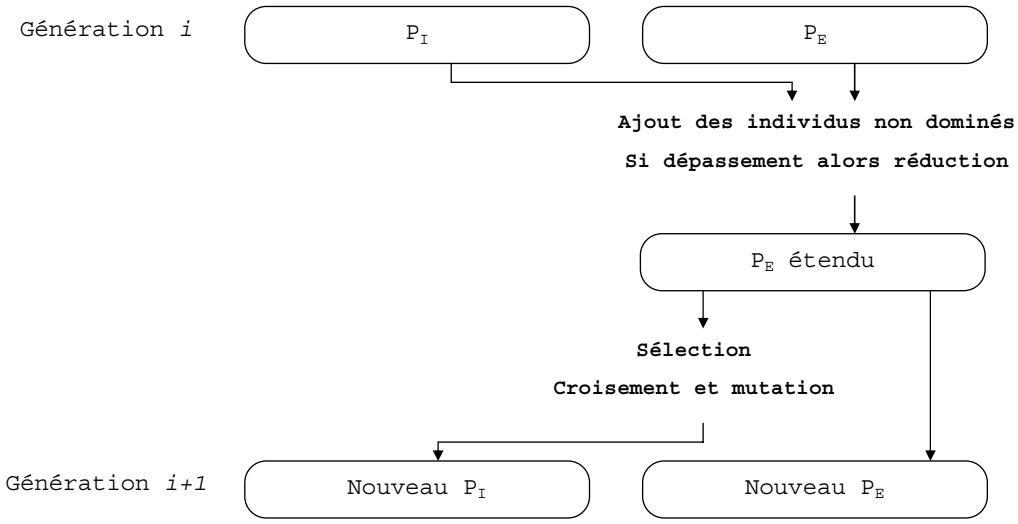


Figure 15 : Schéma de fonctionnement de PESA

Le paramètre *squeeze_factor* est égal au nombre d'individus qui appartiennent au même hypercube. Il est utilisé comme fitness des individus qui appartiennent à cette zone. Par exemple, dans la figure ci-dessous, les points *B* et *D* ont un *squeeze_factor* égal à *1* et *C* a un *squeeze_factor* égal à *3*. Ce paramètre est utilisé pour la sélection ainsi que pour la mise à jour de l'archive alors que dans PAES, la mesure d'encombrement n'est utilisée que pour la mise à jour de l'archive.

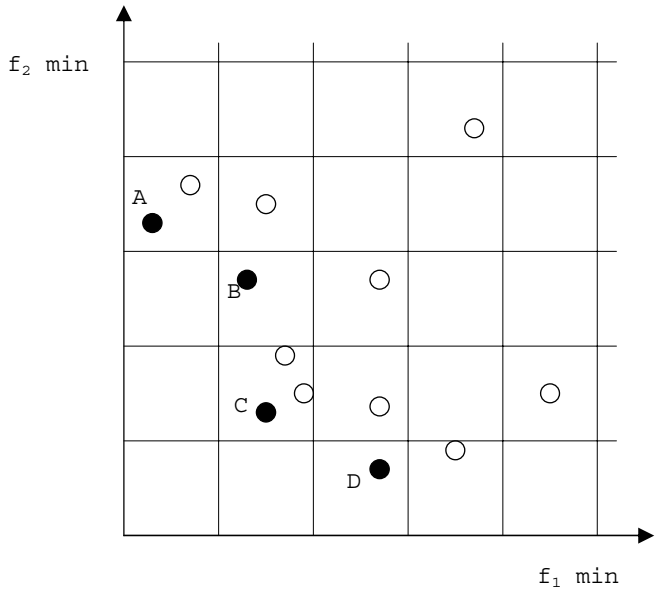


Figure 16 : Mesure du *squeeze_factor* de PESA

Lors de la sélection par tournoi, le candidat choisi sera celui qui a le plus petit facteur d'encombrement. Ainsi les recherches seront orientées vers des zones de la frontière de Pareto qui sont le moins représentées dans la population courante.

Les auteurs comparent PESA à SPEA et à PAES sur les trois premières fonctions de test de Deb [Deb 1999]. Ils trouvent que PESA est la meilleure des trois méthodes en pourcentage de solutions sur la frontière de Pareto avec un nombre limité d'itérations.

Critique

Les critiques faites précédemment sur la méthode de crowding de PAES peuvent être réitérées pour PESA.

La différence essentielle par rapport à PAES est que la sélection est basée sur la mesure d'encombrement de l'espace des objectifs. Si cela permet une bonne répartition des individus dans l'espace d'état, en contre-partie cela augmente la dépendance de l'efficacité de la méthode par rapport au facteur de discrétisation de l'espace.

2.7.3.4 NSGA II

Dans cette deuxième version de NSGA [Deb 2000], l'auteur tente de résoudre toutes les critiques faites sur NSGA : complexité, non élitisme et utilisation du sharing.

La complexité de l'algorithme NSGA est notamment due à la procédure de création des différentes frontières. Pour diminuer la complexité de calcul de NSGA, Deb propose une modification de la procédure de tri de la population en plusieurs frontières.

Algorithme

a) Pour chaque solution il calcule deux paramètres :

   n_i, le nombre de solutions qui dominent la solution i et

   S_i, l'ensemble des solutions que la solution i domine.

   Le calcul de ces deux paramètres demande $O(k.N^2)$ comparaisons.

b) Il identifie les points tels que $n_i = 0$. Ces points forment la frontière F_1.

c) Pour chaque élément i de F_1, il visite l'ensemble S_i, et il retranche 1 au n_j de chaque élément $_j$ de S_i. Cette action demande $O(N)$ calculs.

d) Il recommence en c) jusqu'à ce que tous les points soient traités.

Au total, cet algorithme a une complexité de $O(k.N^2) + O(N) \approx O(k.N^2)$.

L'autre critique sur NSGA est l'utilisation du sharing, méthode qui exige le réglage d'un ou plusieurs paramètre(s) et qui est également forte consommatrice de calculs. Dans NSGA II, Kalyanmoy Deb remplace la fonction de sharing par une fonction de crowding. Il attribue deux caractéristiques à chaque individu :

- i_{rank} représente le rang de non domination de l'individu. Cette caractéristique dépend de la frontière à laquelle appartient l'individu.
- $i_{distance}$ représente la distance de crowding de l'individu et permet d'estimer la densité de la population autour de lui.

Pour estimer la densité au voisinage d'une solution i, il calcule la distance moyenne sur chaque objectif, entre les deux points les plus proches situés de part et d'autre de la solution. Cette quantité appelée $i_{distance}$ sert d'estimateur de taille du plus large hypercube incluant le point i sans inclure un autre point de la population. L'algorithme de calcul est de complexité $O(k.N.\log(N))$. Cette distance de crowding va être utilisée pour guider le processus de sélection.

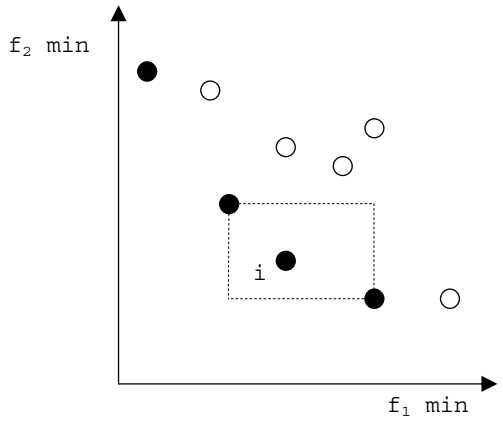


Figure 17 : Calcul de la distance de crowding dans NSGA

Pour répondre à la critique de non élitisme, Deb utilise dans cette méthode une sélection par tournoi et modifie la procédure de passage entre deux générations. Si deux solutions sont sélectionnées pour participer au tournoi, la solution de plus bas rang i_{rank} sera retenue. Mais si les deux rangs sont identiques, il est préférable d'utiliser le point situé dans une région dépeuplée, c'est-à-dire avec une valeur $i_{distance}$ importante. Deb définit sa notion de préférence entre deux solutions de la façon suivante :

$$i \geq j \text{ si } (i_{rank} < j_{rank}) \text{ ou } ((i_{rank} = j_{rank}) \text{ et } (i_{distance} > j_{distance})) \qquad 22)$$

Algorithme de NSGA II

a) La première génération P_0 de taille N est créée aléatoirement.

b) Sélection par tournoi en utilisant la préférence définie ci-dessus et application des opérateurs de modification pour créer un ensemble d'enfants Q_t de taille N.

c) On mélange P_t et Q_t : $R_t = P_t \cup Q_t$.

d) Calcul de toutes les frontières F_i de R_t et ajout dans P_{t+1} jusqu'à ce que la taille de P_{t+1} soit égale à N.

   Les éléments de la dernière frontière calculée sont triés en fonction de la préférence citée ci-dessus. Seuls les éléments permettant à P_{t+1} d'atteindre la taille N sont sélectionnés.

e) On recommence en b)

Les auteurs effectuent une comparaison avec PAES, ils en déduisent que NSGA II est meilleure pour la répartition des individus sur la frontière de Pareto.

Critique

Cette nouvelle version de NSGA a permis de réduire la complexité de l'algorithme à $O(k.N^2)$, de créer une méthode plus élitiste et de supprimer les paramètres de sharing.

2.7.3.5 PESA II : Region-Based Selection

PESA II est une nouvelle technique de sélection basée sur l'utilisation d'hypercubes dans l'espace des objectifs [Corne 2001]. Au lieu d'effectuer une sélection en fonction de la fitness des individus comme dans PESA, cette méthode effectue une sélection par rapport aux hypercubes occupés par au moins un individu. Après avoir sélectionné l'hypercube, on choisit aléatoirement l'individu dans l'hypercube. Cette méthode se montre plus efficace à repartir les solutions sur la frontière de Pareto. Cela est dû à sa capacité de choisir avec une plus grande probabilité que le tournoi classique, des individus situés dans des zones désertiques.

Par exemple dans la figure ci-dessous, les 10 points sont répartis dans 6 cubes. Si l'on considère un tournoi binaire alors la probabilité de sélectionner la solution *A* est :

- dans PESA : $1 - (9/10)^2 = 0,19$
- dans PESA II : $1 - (5/6)^2 \approx 0,31$

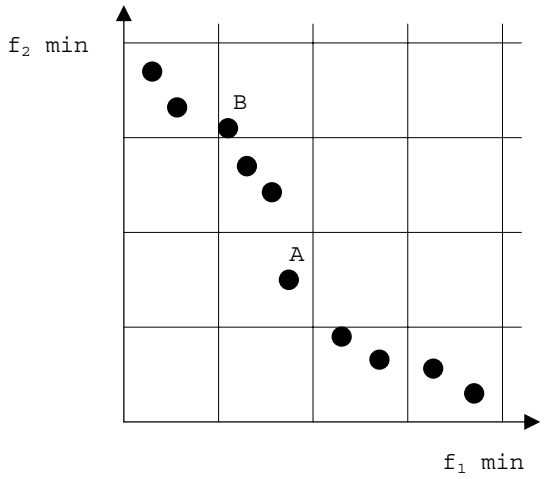


Figure 18 : Exemple de sélection par tournoi hypercube

Critique

PESA II a permis de faire évoluer positivement la sélection de manière à privilégier les zones de l'espace les moins encombrées. Mais, même si cette technique de crowding basée sur un découpage de l'espace est très supérieure en temps de calcul au sharing, elle possède ses propres difficultés. Dans la figure ci-dessus, le point *B* a la même probabilité de sélection que *A* alors que *B* n'est pas situé dans une zone désertique[43]. On peut aussi remarquer que si le découpage avait été de *4* cubes au lieu de *16* alors la probabilité de sélectionner la solution *A* serait passée de *0,31* à *0,89*. Cela montre la très forte influence de la discrétisation de l'espace sur cette méthode.

[43] Si l'on regarde la frontière de Pareto dans son ensemble.

2.7.3.6 Micro-Genetic Algorithm (micro-GA)

Coello trouve que les recherches actuelles n'accordent pas assez d'importance à l'efficience des méthodes d'optimisation multiobjectifs. Dans [Coello 2001], il propose une méthode basée sur une population avec un nombre très faible d'individus. Cette technique se base sur les résultats théoriques obtenus par Goldberg [Goldberg 1989a], selon lesquels une taille de population très faible suffit à obtenir une convergence indépendamment de la longueur du chromosome. Le mécanisme suggéré par Goldberg est d'utiliser une population très petite et d'appliquer les opérateurs génétiques jusqu'à obtention d'une convergence (identifiée par une perte de diversité génétique). Ensuite on crée une nouvelle population en copiant les meilleurs individus de la population précédente, le reste de la population étant généré aléatoirement.

Coello applique ce mécanisme aux problèmes d'optimisation multiobjectifs en utilisant un algorithme génétique avec une petite taille de population associée à une archive et une heuristique de distribution géographique[44]. Il définit une population extérieure divisée en deux parties : une partie remplaçable et une partie non remplaçable. La portion non remplaçable ne change pas durant le processus de modification et sert à maintenir la diversité. Elle ne sera mise à jour que lorsque le micro algorithme génétique aura convergé. La portion remplaçable est totalement modifiée à intervalle régulier. Ce dernier est défini en nombre de cycles du micro GA.

Au début de l'algorithme du micro GA, la population est constituée à l'aide d'individus sélectionnés aléatoirement dans la population externe. Ensuite l'algorithme se déroule de manière classique. En fin de cycle, lorsque la population du micro GA a perdu sa diversité, deux individus non dominés sont sélectionnés pour mettre à jour l'archive. L'approche utilisée est similaire à celle de PAES. Ensuite ces deux mêmes individus sont comparés à deux individus de la partie non remplaçable. Si l'un des deux premiers domine l'un des deux autres alors il le remplace dans la partie non remplaçable.

[44] Cette heuristique permet de répartir uniformément les individus sur la frontière de Pareto. L'auteur utilise la technique de crowding proposée par Knowles et Corne pour la méthode PAES.

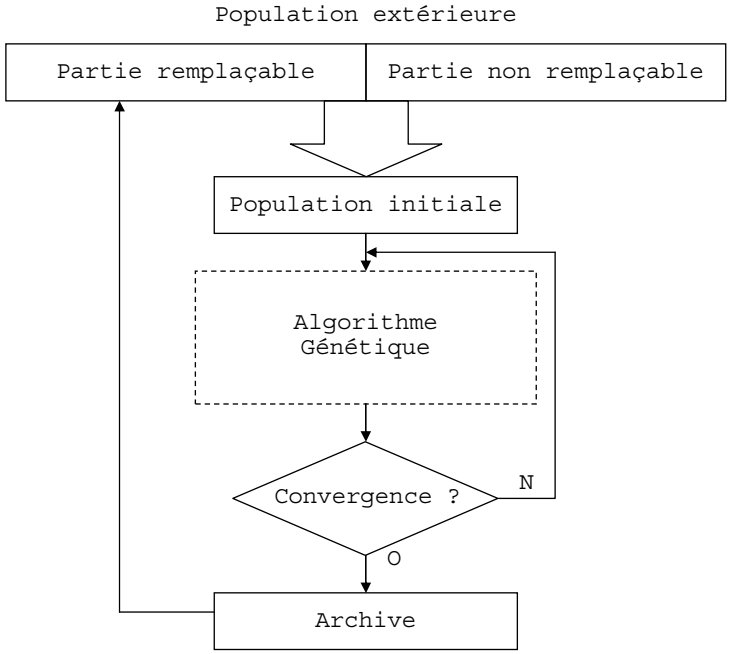


Figure 19 : Schéma de fonctionnement de micro GA

Les tests effectués par l'auteur montrent que cette approche est capable de converger plus rapidement vers la surface de Pareto (en terme de temps CPU). Mais dans le cas de fonctions contraintes la méthode a été moins bonne que NSGA II. Dans quelques cas, cette méthode produit une meilleure distribution des points sur la surface de Pareto.

Critique

L'inconvénient majeur de cette méthode est le nombre de paramètres supplémentaires que l'utilisateur va devoir régler : taille de la population extérieure, taille de la population initiale, taille de l'archive, pourcentage de la partie non remplaçable, intervalle de mise à jour de la partie remplaçable.

2.7.3.7 Memetic[45]-PAES

Cette méthode proposée par Knowles et Corne est une extension de la recherche locale (*1*+*1*)-PAES combinée avec l'utilisation d'une population *P* [Knowles and Corne 2000c]. Périodiquement, celle-ci utilise le croisement pour combiner les optima locaux trouvés par la procédure (*1*+*1*)-PAES. La mise à jour de l'archive est un peu plus compliquée que pour PAES.

Dans PAES, l'archive sert à mémoriser les solutions non dominées trouvées jusque là et d'ensemble de comparaison pour évaluer la dominance des nouveaux individus. Pour effectuer cela M-PAES a besoin de deux archives car les phases de recherche locale et globale doivent être indépendantes :

[45] Le terme "memetic algorithms" a été utilisé pour la première fois par Pablo Moscato [Moscato 1989]. Les "memetic algorithms" combinent une heuristique de recherche locale avec des opérateurs de croisement sur une population.

une archive globale *G* pour mémoriser un ensemble de solutions non dominées et une archive locale *H* utilisée comme échantillon de comparaison lors des recherche locales.

Algorithme

M-PAES se déroule en deux phases : une première phase de recherche locale suivie d'une phase de combinaison.

Au début de chaque génération, *H* est vidée puis remplie avec des solutions de *G* qui ne dominent pas la solution courante *c*. Ensuite, *H* est utilisée comme dans (*1+1*)-PAES. La seule différence entre la recherche locale de PAES et M-PAES est la façon dont se termine la recherche. M-PAES s'arrête lorsque le nombre *l_opt* de mouvements de recherche locale est atteint ou lorsque le nombre d'échecs *l_fails* est atteint. Ce dernier est incrémenté chaque fois que le mutant obtenu est dominé par la solution courante et remis à zéro à chaque mouvement de la solution.

Lors de la phase de combinaison, deux parents sont choisis aléatoirement parmi $P \cup G$ et l'enfant généré sera accepté s'il domine certains membres de *G* ou s'il est situé dans une région peu encombrée à condition qu'il ne soit dominé par aucun membre de *G*. Cette procédure est répétée jusqu'à ce qu'un enfant soit accepté, pendant un nombre maximum de fois *trials_max*.

Les tests effectués par les auteurs montrent que M-PAES est supérieure dans tous les cas à (*1+1*)-PAES. Mais la comparaison avec SPEA a produit des résultats équivalents bien qu'il soit difficile de comparer ces deux algorithmes.

Critique

Bien que l'algorithme soit plus complexe que les précédents, cette nouvelle méthode de Knowles et Corne apporte des perspectives intéressantes. L'utilisation combinée d'une recherche locale et d'une population permet d'associer la capacité exploratoire locale à une capacité d'exploration globale par deux mécanismes indépendants.

Le point négatif de cette méthode est le processus de terminaison de l'algorithme de recherche locale. Comment peut-on justifier le choix des valeurs de *l_opt* et de *l_fails* d'un point de vue décisionnel ?

La même critique pourrait être faite pour le paramètre *trials_max* mais nous sommes dans le cadre d'un algorithme élitiste. Donc le fait de refuser la création d'un enfant moins bon que ses parents nous oblige à imposer un nombre maximal de tentatives car les parents peuvent être totalement incompatibles, c'est-à-dire qu'ils ne généreront jamais un enfant meilleur qu'eux.

2.8 Conclusion sur les méthodes d'optimisation multiobjectifs

Dans cette section, nous présentons tout d'abord les difficultés rencontrées par le processus d'optimisation des méthodes multiobjectifs, puis nous discutons de la difficulté de mise en œuvre de ces méthodes. Nous mettrons l'accent sur les points suivants : le nombre de paramètres et le critère d'arrêt de l'algorithme.

2.8.1 Difficultés des méthodes d'optimisation multiobjectifs

Un processus d'optimisation multiobjectif doit résoudre les deux tâches suivantes :

- guider le processus de recherche vers la frontière de Pareto,
- maintenir une diversité des solutions pour assurer une bonne répartition sur la frontière de Pareto.

L'accomplissement de ces tâches est très délicat car les difficultés rencontrées dans un problème multiobjectif sont identiques à celles d'un problème simple objectif mais elles sont amplifiées par la présence d'objectifs dépendants les uns des autres.

2.8.1.1 Guider le processus de recherche vers la frontière de Pareto

Le processus de recherche est souvent ralenti ou totalement dérouté par des fonctions possédant une des caractéristiques suivantes : *multimodalité*, *isolation d'un optimum* ou *tromperie*.

La multimodalité

Une fonction est dite multimodale si elle possède plusieurs d'optima globaux (Figure 20). Dès lors, chaque optimum exerce sur les individus d'une population une attraction différente qui peut piéger le processus de convergence de l'algorithme. Ce problème peut être évité en utilisant une technique de répartition (Partie I.3.5) des individus de type sharing ou crowding [Mahfoud 1995].

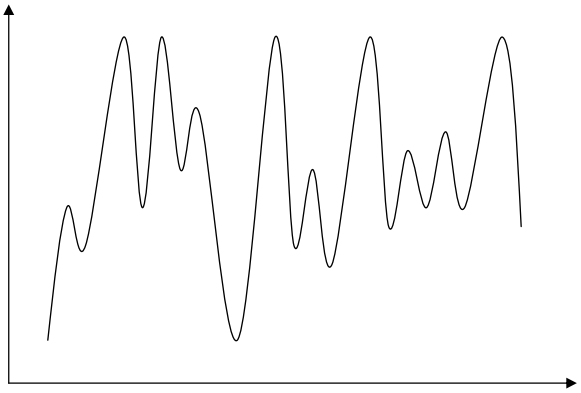


Figure 20 : Une fonction multimodale

L'isolation d'un optimum

Il existe des problèmes dans lesquels un optimum peut être entouré de grandes zones pratiquement plates. Cet optimum se trouve alors isolé car l'espace de recherche qui l'entoure ne peut pas guider vers lui les individus de la population.

Pour toutes les méthodes présentées ci-dessus, il est très difficile de garantir au décideur que ce type d'optimum peut être trouvé. Par contre les méthodes utilisant une population externe comme archive (Partie I.2.7.3) semblent plus aptes à maintenir un optimum isolé que celles qui n'en utilisent pas.

La tromperie

Un problème est trompeur (Figure 21) lorsqu'il guide la convergence vers une zone non optimale de la fonction.

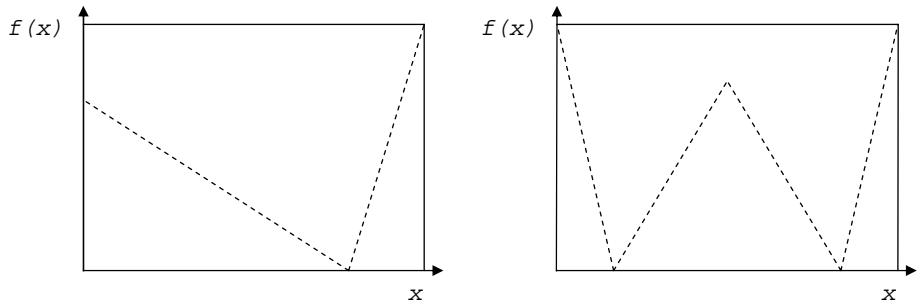


Figure 21 : Problèmes trompeurs

Dans les deux problèmes ci-dessus, nous voyons que la probabilité de choisir aléatoirement un point dans une zone sous-optimale est très grande. Donc, dans un premier temps, les méthodes ont une tendance à converger vers des optima locaux. Par la suite, le processus de mutation entretient ce phénomène de tromperie.

Pour éviter cette tromperie, Deb et Goldberg recommandent l'utilisation de techniques de répartition des individus en niches [Goldberg and Deb 1992]. Ils établissent également que le choix d'une taille appropriée de la population est primordial pour éviter la tromperie.

2.8.1.2 Maintenir la diversité sur la frontière de Pareto

La difficulté à maintenir une bonne répartition des solutions sur la frontière de Pareto résulte principalement des caractéristiques suivantes : convexité ou non convexité de la frontière de Pareto, discontinuité de cette frontière et non uniformité de la distribution.

Convexité et non convexité

Certains problèmes ont une frontière de Pareto non convexe. Les méthodes dont le calcul de la fitness est basé sur le nombre d'individus dominés (MOGA Partie I.2.7.2.1, SPEA Partie I.2.7.3.1) vont être moins efficaces.

Discontinuité

Si une frontière de Pareto est discontinue, nous retrouvons le même principe que pour une fonction multimodale. Les différentes parties de cette frontière vont exercer proportionnellement à leur taille, une attraction plus ou moins importante sur les individus d'une population. Certaines d'entre elles pourront donc ne pas être découvertes.

Les méthodes basées sur une population génétique sont plus sensibles à ce phénomène que les méthodes utilisant des stratégies d'évolution.

La non uniformité de répartition sur la frontière

Les solutions sur la frontière de Pareto peuvent ne pas être réparties uniformément. La raison principale vient du choix des fonctions objectifs. Par exemple, si une des fonctions objectifs est multimodale, elle va influencer de manière très différente la répartition des solutions sur la frontière de Pareto.

2.8.2 La mise en œuvre

2.8.2.1 Le paramétrage

La mise en œuvre de la plupart des méthodes d'optimisation multiobjectifs présentées ci-dessus exige en priorité de maîtriser le fonctionnement d'un algorithme génétique et de comprendre l'interaction de ses différents paramètres.

La maîtrise du fonctionnement d'un algorithme génétique permet à l'utilisateur, à partir d'un problème réel, de répondre aux questions suivantes : Quel génotype ? Quelle notation ? Quelle fonction de sélection ? Quels types de croisement et de mutation ? Quelle heuristique de partage (éventuellement) ? Quel critère d'arrêt ?

Si la compréhension des interactions entre paramètres peut être aisément appréhendée sur des exemples simples, elle reste une question délicate pour les nouveaux venus dans ce domaine. Car déterminer la taille de la population, le taux de croisement et le taux de mutation n'est pas une chose aisée. Si en plus on ajoute une heuristique de partage, il faut également prendre en compte les paramètres de définition du voisinage.

Les méthodes basées sur une stratégie d'évolution sont algorithmiquement plus simples. Mais l'usage systématique d'une population externe pour maintenir les meilleurs individus exige la détermination de stratégies de mise à jour et de réutilisation basées sur une heuristique de remplacement géographique qui suppose la définition d'un maillage de l'espace de recherche.

2.8.2.2 La définition d'un critère d'arrêt

La définition d'un critère d'arrêt est une difficulté supplémentaire.

Le critère le plus usité pour stopper la recherche d'un algorithme génétique est la perte de diversité génétique. Or celui-ci n'a plus de sens dans des méthodes qui maintiennent la diversité génétique de la population pour permettre une recherche plus efficace. Peu d'auteurs expriment clairement leur critère d'arrêt de l'algorithme. Mais en pratique, cela n'est pas critiquable car l'utilisation de ces méthodes par un décideur pourra se faire de manière interactive.

Dans la méthode Memetic PAES (Partie I.2.7.3.7), basée sur une stratégie d'évolution, les auteurs définissent un critère d'arrêt basé sur le nombre d'échecs consécutifs de la recherche locale. Dans ce cas il est difficile pour un décideur d'établir une relation de cause à effet entre le nombre d'échecs et la qualité du résultat.

Nous verrons que dans la méthode que nous avons réalisée (Partie II), nous nous sommes attachés à réduire ces deux dernières difficultés tout en conservant les qualités de convergence et de maintien de la diversité du processus d'optimisation.

Le tableau récapitulatif suivant regroupe les caractéristiques essentielles des méthodes vues précédemment. Il met particulièrement l'accent sur le grand nombre de paramètres utilisés dans la plupart de ces méthodes. Le lecteur pourra appréhender la difficulté de mise en œuvre de ces méthodes car la relation entre la valeur d'un paramètre et son action sur la résolution du problème est très difficile à contrôler sans une parfaite connaissance de la méthode employée. De plus, beaucoup de ces paramètres ont peu de sens pour le décideur. Dans la méthode que nous avons développée, un accent tout particulier est mis sur un paramétrage simple et compréhensible pour un non initié.

Méthode	Action	Sharing (d, σ_{share} et α)	Restricted mating (σ_{mate})	Crowding (Si grille : l)	Clustering (minDist)	Population génétique (N, t_c et t_m)	Archive (taille)	Paramètres supplémentaires	Complexité
VEGA	Sélection sur un seul objectif à la fois + mélange		Oui			Oui			
Utilisation de genre	Sélection basée sur le rang de l'individu + notion d'attracteur sexuel		Oui			Oui		c_d, c_w, k_δ, k_w, n_1, n_2	
Méthode non générationnelle	Notation basée sur la technique des classifieurs	Oui				Oui			
MOGA	Notation, calcul du rang + interpolation	EO				Oui			
NSGA	Notation, tri par rapport aux différents degrés de dominance	EE				Oui		Fitness factice	$k.N^3$
NPGA	Sélection, tournoi de dominance sur un sous-ensemble de la pop	EE				Oui		t_{dom}	
SPEA	Notation relative au nombre d'individus dominés				Oui	Oui	Oui		
PAES	Stratégie d'évolution (1+1)			Grille, EO			Oui		$k.N^2$
PESA	Sélection, tournoi en fonction de la mesure d'encombrement			Grille, EO		Oui	Oui	squeeze factor	
NSGA II	Notation, idem NSGA. Sélection, tournoi classique + préférence en fonction de degré d'encombrement de l'espace			EO		Oui	Oui		$k.N^2$
PESA II	Sélection par rapport aux hypercubes			Grille, EO		Oui	Oui		
Micro GA	AG avec une population très réduite					Oui	Oui	pop_{ext}, $\%_{rempl}$ et $inter_{maj}$	
Memetic-PAES	PAES avec une population et 2 archives			Grille, EO		Oui	Oui (2)	l_opt, l_fails et trials_max	

Tableau 5 : Récapitulatif des méthodes d'optimisation multiobjectifs

Légende : EO : appliqué dans l'espace des objectifs, EE : appliqué dans l'espace des états.

2.9 Les problèmes de test

Dans cette section, nous présentons quelques fonctions de test qui peuvent être utilisées pour comparer ces méthodes.

Dans [Deb 1999], l'auteur propose un ensemble de caractéristiques spécifiques pour construire des problèmes de test pour des méthodes d'optimisation multiobjectifs. Il propose une ensemble de six fonctions et effectue un test sur plusieurs des méthodes présentées ci-dessus [Zitzler and Deb 1999].

La même année, David Van Veldhuizen propose dans [Van Veldhuizen 1999], un récapitulatif des fonctions utilisées pour tester les méthodes d'optimisation multiobjectifs.

Il présente les caractéristiques de chaque fonction dans le tableau suivant :

Nom de la Fonction	Génotype						Phénotype				
	Continue	Discontinue	Symétrique	Nombre de variables	Nombre d'objectifs	Nombre de contraintes	Géométrie	Continue	Discontinue	Concave	Convexe
Viennet 1	X		X	2	3	2	Surface	X			
Viennet 2	X			2	3	2	Surface	X			
Viennet 3		X		2	3	2	Courbe	X			
Viennet 4		X		2	3	2+3	Surface	X			

Figure 22 : Exemple de fonctions de test

Le problème n°1 proposé par Viennet est le suivant :

$$f = \min (f_1(x, y), f_2(x, y), f_3(x, y)) \text{ avec } -4 \le x, y \le 4 \text{ et}$$

$$f_1(x, y) = x^2 + (y-1)^2$$

$$f_2(x, y) = x^2 + (y+1)^2$$

$$f_3(x, y) = (x-1)^2 + y^2 + 2$$

Le problème n°2 est le suivant :

$$f = \min (f_1(x, y), f_2(x, y), f_3(x, y)) \text{ avec } -3 \le x, y \le 3 \text{ et}$$

$$f_1(x, y) = (x-2)^2/2 + (y+1)^2/13 + 3$$

$$f_2(x, y) = (x+y-3)^2/36 + (-x+y+2)^2/8 - 17$$

$$f_3(x, y) = (x+2y-1)^2/175 + (2y-x)^2/17 - 13$$

Chapitre 3.
Les problèmes d'optimisation en environnement dynamique

3.1 Définition

Un problème d'optimisation en environnement dynamique est un problème dont la fonction objectif change au cours du temps. Cette dynamique implique une nouvelle problématique pour les méthodes d'optimisation.

3.2 Problématique

Pour résoudre ce type de problème, les méthodes d'optimisation doivent être capables non seulement de trouver l'optimum de la fonction mais également de suivre cet optimum au cours du temps. Deux stratégies différentes peuvent être envisagées pour maintenir cet optimum :

- savoir retrouver très rapidement l'optimum lorsqu'il a été modifié suite à un changement,
- suivre effectivement l'optimum lors des changements.

Dans un système dynamique, la qualité de la solution trouvée sera un critère moins déterminant que pour les méthodes d'optimisation en environnement stationnaire. Par contre, la robustesse de ces méthodes sera évaluée par rapport à leurs capacités à :

- détecter un changement,
- fournir une réponse appropriée.

Nous dirons qu'une méthode a une réponse appropriée si **son approche est discriminante**, c'est-à-dire, si sa réponse au changement d'environnement prend en compte la zone de l'espace dans laquelle a eu lieu cet évènement.

3.3 Classification des méthodes

Une première classification des algorithmes évolutionnaires d'optimisation de problèmes dynamiques a été proposée par Branke [Branke 2001], l'auteur regroupe les méthodes en quatre classes :

- Les méthodes qui réagissent au changement. Ces dernières sont utilisées de manière classique mais lorsqu'on détecte un changement, des actions extérieures permettent d'augmenter la diversité de la population et facilitent ainsi la recherche du nouvel optimum.
- Les méthodes qui maintiennent la diversité. En conservant une bonne répartition des individus dans l'espace, on espère pouvoir réagir plus efficacement au changement.
- Les méthodes qui utilisent une mémoire permettant de conserver les optima passés. Mais elles ne sont utilisables que dans les cas pour lesquels l'optimum a un mouvement périodique.
- Les méthodes qui utilisent plusieurs populations réparties de manière à suivre les optima locaux et à rechercher le nouvel optimum.

3.4 Les méthodes réactives

3.4.1 Heuristique

Le principe général des méthodes réactives est d'effectuer une action extérieure suite à la détection d'un changement de l'optimum. En général, cette action a pour but d'augmenter la diversité génétique de la population. Cela va permettre à l'algorithme de retrouver sa capacité de recherche. Les méthodes basées sur l'utilisation de populations d'individus perdent de la diversité lorsqu'elles convergent vers un optimum. Ce manque de diversité devient handicapant lorsqu'il y a perte de l'optimum et que l'algorithme doit recommencer sa recherche. Donc, en augmentant à nouveau la diversité de la population, le processus de recherche est relancé.

3.4.2 Les techniques

3.4.2.1 Hypermutation

Après la détection d'un changement, cette technique conserve la population à l'identique mais elle augmente énormément le taux de mutation pendant quelques générations de façon à élever la diversité de la population [Cobb 1990]. Dans son article Helen Cobb démontre les effets positifs d'un taux de mutation variable par rapport à un taux fixe et en déduit que ce taux doit refléter le

niveau de stationnarité de l'environnement[46] et varier proportionnellement à l'importance de changement de l'environnement.

Elle effectue des tests sur une fonction unimodale dépendante du nombre de générations du type :

$$\sin(\alpha.génération) + 1 \qquad 23)$$

Lors de ses expériences, elle utilise comme paramètre de performance la moyenne sur dix tests du meilleur individu à chaque génération.

3.4.2.2 Variable Local Search (VLS)

Dans [Varak 1997a], l'auteur développe une méthode un peu différente de celle d'Helen Cobb. L'augmentation du taux de mutation se fait graduellement. Au départ ce taux est bas, mais si pendant un certain nombre de générations il n'y a pas d'amélioration des fitness, le taux de mutation est augmenté. Pour mettre en œuvre une mutation graduelle, Varak utilise un opérateur de décalage qui détermine à partir du bit de poids faible le nombre de bits qui peuvent être modifiés par la mutation.

$$\pm 2^{bit}\text{-}1 \text{ avec } bit \text{ le paramètre de réglage} \qquad 24)$$

Dans [Varak 1997b], l'auteur propose une stratégie pour adapter le taux de mutation.

3.4.3 Discussion

Les deux méthodes ci-dessus sont très simples à mettre en œuvre, mais elles demandent de répondre aux questions suivantes :

1) Comment détecter le changement ?

   La détection du changement est une étape très importante car la réponse sera d'autant plus efficace si la détection s'effectue relativement tôt, et, en cas d'échec de la détection ces méthodes ne fonctionneront pas correctement. Les techniques de détection les plus simples à mettre en œuvre sont des techniques d'études statistiques. Mais le bon fonctionnement de ces méthodes demande une bonne connaissance a priori du problème par le décideur pour pouvoir régler les paramètres.

2) Comment déterminer la fonction de variation du taux de mutation ?

   Pour une majorité des problèmes, il n'existe pas de règles permettant de fixer le taux optimal de mutation fixe, alors déterminer un taux variable n'est réalisable que pour des problèmes biens définis. De plus, l'utilisation d'un taux de mutation automatique peut induire des difficultés lorsqu'il n'y a pas ou peu de variation d'environnement.

[46] Le taux sera bas en situation stationnaire et élevé en situation non stationnaire.

De plus en introduisant un taux de mutation trop important ces méthodes risquent de dégrader le processus de convergence.

Leur mise en place n'implique pas un surcoût en temps de calcul très important. Mais la technique de détection du changement peut nécessiter plus de calculs que la réponse elle-même.

Enfin, la réponse de ces méthodes à un changement n'est pas discriminante. Elle s'effectue sur l'ensemble de la population sans prendre en compte la zone de l'espace dans laquelle a eu lieu le changement. Pour preuve, à chaque détection d'un changement, la fitness de tous les individus doit être recalculée.

Les méthodes que nous allons voir maintenant adoptent une stratégie différente.

3.5 Les méthodes préventives de maintien de la diversité

3.5.1 Heuristique

Ces méthodes tentent de maintenir la diversité de la population en espérant que lors d'une modification de la fonction objectif, la répartition des individus dans l'espace d'état permettra de retrouver plus rapidement le nouvel optimum.

3.5.2 Des techniques

3.5.2.1 Random immigrants

Cette méthode a été proposée par Grefenstette [Grefenstette 1992]. Une partie[47] des individus de la population est remplacée par des individus générés aléatoirement. Cette étape est ajoutée après l'étape de modification des individus par l'algorithme génétique. Sur un problème off-line particulier, Grefenstette trouve que le taux optimal pour répondre au changement d'environnement est de 30%. Même si ce taux doit être pris avec prudence, il montre que le taux de remplacement utilisé va être beaucoup plus important que le taux de mutation d'un algorithme génétique classique.

L'avantage de cette méthode par rapport aux précédentes est qu'elle permet d'introduire de la diversité tout en diminuant le risque de dégradation du processus de convergence.

3.5.2.2 Sharing

La technique de sharing, appelée **heuristique de partage**, a été proposée par Goldberg [Goldberg 1989]. Le but de cette technique est d'éviter le regroupement des individus autour d'un optimum local lors de la recherche de l'optimum global dans un problème multimodal. Sans une telle

[47] Même si Grefenstette ne le précise pas, nous pouvons supposer que ceux sont les plus mauvais individus qui sont remplacés dans la population.

heuristique, la population a tendance à converger vers l'optimum local qui influence le plus tôt une partie significative de la population. Cette convergence entraînant une perte de diversité génétique, l'algorithme est par la suite dans l'impossibilité de trouver l'optimum global. Pour éviter ce phénomène, l'heuristique de partage modifie la note de l'individu proportionnellement à son isolement. En d'autres termes, on préférera un individu *i* isolé avec une note moyenne plutôt qu'un individu *j* avec une très bonne note mais dans une zone très peuplée. Ainsi la probabilité de sélection de *i* sera équivalente ou supérieure à celle de *j*. L'heuristique de partage permet de répartir l'ensemble des individus de la population dans différentes niches et évite ainsi une convergence prématurée.

L'exemple ci-dessous présente le résultat de l'application d'une heuristique de sharing. Nous constatons bien sur la figure de droite que les individus de la population sont répartis sur les trois pics.

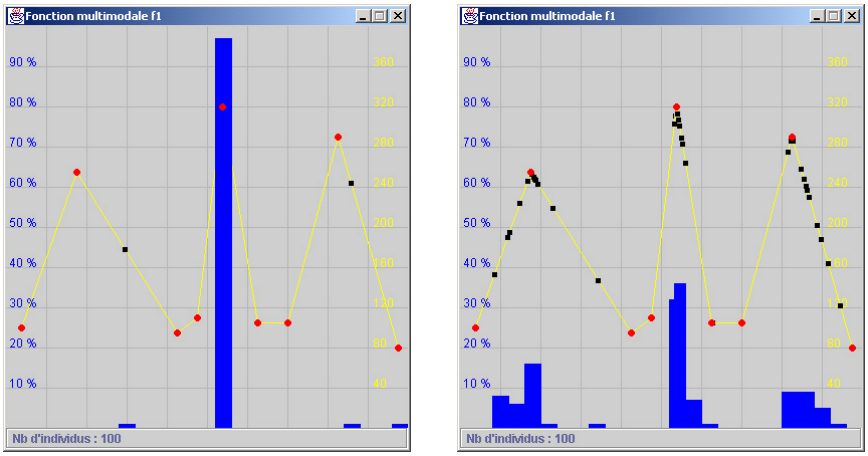


Figure 23 : Exemple de l'utilisation du sharing

L'application du sharing suppose la définition d'une fonction de distance $d(x_i, x_j)$, et la fixation d'un seuil de voisinage σ_{share}. On peut également définir un valeur d'influence α qui amplifiera ou diminuera l'influence de la proximité de *j* par rapport à *i*.

La fonction de fitness calculée avec l'heuristique de partage est :

$$f_i' = \frac{f_i}{m_i} \text{ avec} \qquad 25)$$

$$m_i = \sum_{j=1}^{j=N} S(d(x_i,x_j)) \text{ et} \qquad 26)$$

$$S(d) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} & \text{si } d > \sigma_{share} \\ 0 & \text{sinon} \end{cases} \qquad 27)$$

Dans [Goldberg & Richardson 1987], les auteurs proposent une extension pour des problèmes d'optimisation multidimensionnelle.

En 1991 Andersen examine les effets de l'heuristique de partage appliquée sur le génotype et le phénotype pour suivre le mouvement d'un optimum [Andersen 1991]. Il pense que le maintien de la diversité de la population par le sharing permettra un suivi plus efficace. Il conclut que le sharing augmente énormément la capacité d'un algorithme génétique à suivre un optimum à condition que l'environnement change doucement.

Critique

L'utilisation de l'heuristique de partage suppose que l'on puisse définir une notion de distance entre individus. Dans [Largeron et Auray], les auteurs proposent un ensemble de mesures de proximité dans le cas de variables quantitatives et qualitatives nominales ou ordinales.

Cette technique demande un temps de calcul supplémentaire important d'ordre $O(N^2)$. Pour réduire cette complexité on peut recourir à un **sharing clusterisé** de complexité $O(N.log(N))$ ou à une méthode de **niche dynamique** de complexité $O(q.N)$ [Gan and Warwick, Miller and Shaw 1995] avec q le nombre d'optima locaux de la fonction. L'autre inconvénient de cette technique est le réglage des paramètres σ_{share} et α qui est très dépendant du problème et du type de la mesure de distance définie.

Si la courbe possède de nombreux optima locaux, l'efficacité de cette technique va dépendre du nombre d'individus de la population. En effet, si le nombre d'individus est faible proportionnellement au nombre d'optima locaux, l'heuristique de partage ne pourra pas répartir l'ensemble de la population sur tous les pics. Par exemple, dans la figure ci-dessous, l'heuristique de partage n'arrive pas à distribuer correctement les individus dans l'espace. Ainsi beaucoup de niches petites et moyennes sont oubliées.

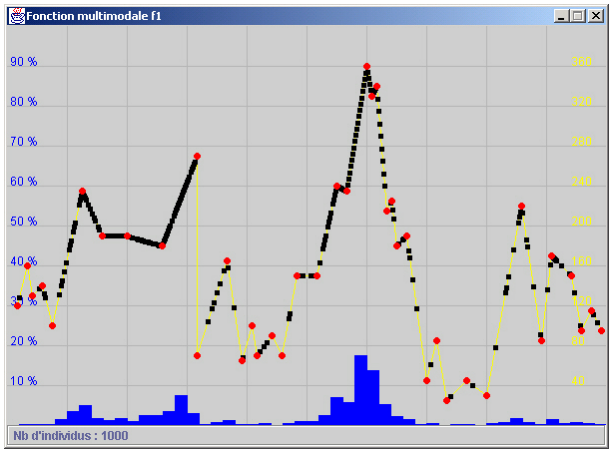


Figure 24 : Sharing sur fonction multimodale

Dans le cadre d'un problème dynamique, la probabilité que le nouvel optimum global soit un ancien optimum local est importante. Donc si ce dernier a été oublié par l'heuristique de partage, le processus de recherche du nouvel optimum va être inefficace. De plus, cette inefficacité est amplifiée par le fait que la vitesse de redistribution dans l'espace d'état des individus est lente lors d'un changement d'environnement.

3.5.2.3 Crowding

Le crowding ou **l'heuristique de remplacement** a été introduit par De Jong [De Jong 1975]. Le principe originel consiste à remplacer des éléments d'une population par d'autres éléments similaires et meilleurs qu'eux. Cette heuristique est une analogie des systèmes vivants en compétition pour une ressource. Dans les problèmes d'optimisation, la ressource est l'optimum à atteindre. Donc, des individus situés dans des zones différentes de l'espace d'état ne sont pas en concurrence pour le même optimum, alors que des individus proches sont en compétition. En remplaçant les individus semblables, le crowding permet de conserver les différentes niches de la population tout en accélérant la convergence.

Dans de nombreux problèmes, cette heuristique est implantée de la façon suivante :

a) On sélectionne deux parents, puis on génère deux enfants par croisement et mutation.

b) Ensuite on forme tous les couples parent-enfant et on calcule la similarité entre le parent et l'enfant.

c) Pour finir, le couple le plus proche est sélectionné et inséré dans la population suivante.

L'algorithme ci-dessus applique le crowding sur le génotype de l'individu et sa complexité est de $O(2.N)$.

Nous avons vu dans le cadre de problèmes multiobjectifs plusieurs utilisations de crowding appliqué sur le phénotype des individus. Nous pouvons également remarquer que parmi ces différentes heuristiques de remplacement, aucune ne prend en compte la similarité entre individus. Au contraire, toutes sont utilisées pour éloigner les solutions les unes des autres. En fait, le terme crowding est conservé car on utilise toujours le principe de remplacement. Cela permet de bien séparer les heuristiques de partage (sharing) et de remplacement (crowding).

Cedeno et Vemuri utilisent cette technique dans le cadre d'un problème dynamique. Ils montrent que cette approche est capable de maintenir différentes solutions dans la population [Cedeno and Vemuri 1997].

Critique

Les différentes recherches ont montré que les résultats en terme de maintien de la diversité et de répartition de la population dans l'espace sont moins bons pour le crowding que pour le sharing dans le cadre de problèmes uniobjectifs ou multimodaux. Par contre l'utilisation du crowding dans

la cadre de problèmes multiobjectifs se montre aussi efficace à répartir les individus sur la frontière de Pareto que le sharing. Mais en terme d'efficacité de calcul, le crowding est dans tous les cas supérieur au sharing.

Le lecteur pourra trouver une analyse complète des techniques de formation de niches dans [Mahfoud 1995].

3.5.2.4 Age des individus

Dans [Ghosh 1998], l'auteur introduit la notion d'âge des individus pour calculer leur fitness effective. Initialement, l'âge a_i d'un individu est égal à zéro, ensuite il est incrémenté de un à chaque génération jusqu'à ce qu'il atteigne une valeur limite.

Ghosh utilise une fonction d'âge $h(a_i)$ conçue de manière à favoriser les individus d'âge moyen par rapport aux individus jeunes ou vieux. Il respecte ainsi l'analogie avec les systèmes vivants.

La fitness effective d'un individu i est une combinaison de la valeur f_i de la fonction objectif et de la fonction d'âge h de l'individu :

$$fe_i = Fit(g(f_i), h(a_i)) \qquad 28)$$

avec g une fonction de modification[48] de la valeur de la fonction objectif

La forme générale de la fonction pour obtenir la fitness effective est la suivante :

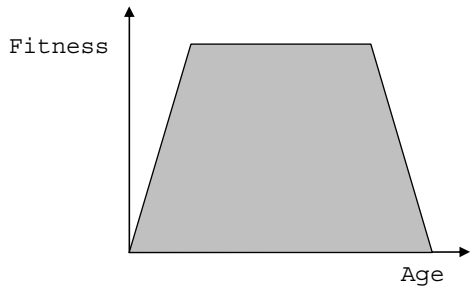


Figure 25 : Variation de la fitness effective en fonction de l'âge

Critique

En diminuant la fitness des jeunes individus, cette heuristique ralentit leur influence. Cela évite une convergence prématurée due à un individu trop fort par rapport au reste de la population. Mais cette action peut aussi avoir un effet négatif, car elle peut faire disparaître l'individu. Cela signifie la perte de bons caractères génétiques. Si l'individu était né suite à un croisement, la perte génétique sera faible car les parents ou des enfants proches existent dans la population. Par contre, si sa naissance était due à une mutation, la probabilité de retrouver l'individu sera très faible.

[48] Par exemple : une fonction de mise à l'échelle.

La limitation de la fitness des individus âgés évite qu'un individu ne subsiste pendant une longue période de temps.

En freinant la convergence rapide d'une part, et en limitant l'existence d'un individu d'autre part, cette heuristique permet d'augmenter la diversité de la population et de répondre au changement de l'environnement, mais aucune comparaison avec les techniques précédentes n'a été faite.

3.5.3 Discussion

A l'opposé des méthodes réactives, ces dernières agissent de manière à prévenir le changement. Ainsi, il n'est pas nécessaire de définir une technique de détection du changement.

Dans certains cas, la réponse de ces méthodes à un changement est plus appropriée que la réponse des méthodes réactives. Si le changement d'optimum s'effectue dans une zone proche d'un ancien optimum local autour duquel s'est formée une niche, la réponse sera rapide et ciblée. Le risque est que le changement s'effectue dans une zone dépeuplée. Le temps de réponse va alors être plus important et la probabilité de détruire des niches situées dans d'autres régions de l'espace n'est pas négligeable. Donc nous ne pouvons pas dire que ces méthodes ont une approche *discriminante*[49].

Ces méthodes préventives sont plus coûteuses en temps de calcul que les méthodes réactives et sont également plus complexes à mettre en œuvre.

3.6 Les méthodes à mémoire

3.6.1 Heuristique

Ces méthodes utilisent une mémoire qui enregistre l'évolution des différentes positions de l'optimum afin de pouvoir les réutiliser un peu plus tard. Bien évidemment, ces méthodes ne peuvent aboutir à de bons résultats que si le changement de position de l'optimum est périodique. Par rapport aux méthodes précédentes que nous avions qualifiées de réactives et préventives, ces méthodes pourrait être dites anticipatives.

La mise en place d'une mémoire dans une population peut se faire d'une manière implicite ou de manière explicite.

Les approches implicites utilisent une représentation différente du génotype afin d'inclure l'historique d'évolution de l'optimum dans son génome.

Les approches explicites utilisent une mémoire externe et nécessite la définition de stratégies d'utilisation de cette mémoire (mise à jour et réutilisation des optima stockés).

3.6.2 Les techniques

Les deux premières techniques présentées ci-après ont une approche implicite alors que les deux dernières ont une approche explicite.

3.6.2.1 La diploïdie

En 1998 Lewis utilise la diploïdie dans le cadre de problèmes dynamiques [Lewis 1998]. Il espère que l'information stockée dans le double chromosome permettra d'assurer la diversité au cours de la simulation et permettra au système de répondre plus vite et de façon plus robuste au changement d'environnement. Il teste différents algorithmes diploïdes avec et sans mécanisme de dominance génétique sur plusieurs problèmes. Il en conclut que cette technique n'est pas assez flexible pour répondre aux problèmes dynamiques. Les tests effectués montrent qu'une méthode utilisant un chromosome haploïde avec une technique d'hypermutation est plus efficace.

3.6.2.2 Représentation multiniveau des gènes

Dans [Dasgupta 1992], l'auteur propose une représentation multiniveau d'un gène. Son algorithme, appelé algorithme génétique structuré, utilise la redondance d'information et une structure hiérarchique des gènes. Dans son modèle, chaque gène a la capacité d'activer ou désactiver les gènes de niveau inférieur. Ainsi un changement dans un gène de haut niveau produit un effet sur le phénotype qui ne pourrait être obtenu par une représentation classique qu'après un certain nombre de modifications. La probabilité de cet évènement est donc très faible.

L'avantage d'une telle représentation est que les gènes de haut niveau peuvent explorer un espace très large alors que les gènes de bas niveau continuent à effectuer une recherche plus locale. Dans un problème dynamique ce type de représentation est donc intrinsèquement meilleur pour retrouver les bons schèmas par rapport à une représentation classique.

Par contre, cette représentation exige plus de mémoire pour stocker le génome d'un individu, et elle est très difficile à mettre en œuvre.

L'approche suivante utilise également une mémoire mais celle-ci est extérieure aux individus.

3.6.2.3 Méthode par expérience

Dans [Xu and Louis 1996], les auteurs présentent une méthode qui mélange les algorithmes génétiques avec un raisonnement par expérience[50] appliqué à un problème de planification de tâches.

Le raisonnement par expérience est basé sur l'idée qu'une décision ou une explication peut être meilleure lorsqu'elle fait référence aux cas déjà trouvés dans le passé. Lorsque le décideur est

[49] Rappel : une approche est dite discriminante, si la réponse au changement d'environnement prend en compte la zone de l'espace dans laquelle a eu lieu cet évènement.
[50] Trad. Case-based reasoning (CBR).

confronté à un problème, il essaie de retrouver le cas le plus similaire dans sa mémoire et il tente ensuite de l'adapter au problème présent.

Pour simuler ce genre de raisonnement dans un algorithme génétique, les auteurs enregistrent le meilleur individu de la population à intervalle régulier. Après un changement, la population est réinitialisée, une partie de 5 à 10% est remplacée par les informations stockées dans la mémoire et le reste des individus est initialisé aléatoirement.

Après leurs expériences, Xu et Louis constatent qu'utiliser une population de sauvegarde trop grande ou injecter un pourcentage trop important d'anciennes solutions est pénalisant pour la convergence. Ils constatent également que leur méthode échoue si l'environnement se modifie trop souvent.

Préalablement Ramsey avait également couplé un raisonnement par expérience avec un algorithme génétique incorporé dans un système d'apprentissage [Ramsey 1993]. Ce dernier est composé d'un module d'exécution et d'un module d'apprentissage.

Le module d'exécution est constitué d'une base de connaissances active qui représente les meilleures règles trouvées, et d'un moniteur capable d'indiquer au module d'apprentissage un éventuel changement dans l'environnement. Périodiquement, une règle est sélectionnée dans la base du module d'apprentissage pour être ajoutée à la base active.

Le module d'apprentissage est identique à un système de classifieurs. Il possède un ensemble de règles stimulus/réponse qui mettent en correspondance l'état de l'environnement à un certain instant t avec la réponse la mieux adaptée. A chaque détection de changement d'environnement par le moniteur, l'algorithme génétique est relancé pour découvrir de nouvelles règles. La population initiale est composée pour moitié des meilleures règles extraites de la base active, d'un quart de règles de la population précédente, d'un huitième de règles générées aléatoirement et d'un huitième de règles par défaut.

Critique

L'avantage d'utiliser une base de connaissances externe au système d'apprentissage est double. Tout d'abord, elle permet de maintenir une stabilité comportementale par rapport à l'environnement. Si ce dernier ne se modifie pas trop souvent, cette approche est bénéfique. Enfin, cette base permet de réinitialiser en partie l'algorithme génétique sur des connaissances passées. La convergence peut ainsi être accélérée si le changement d'environnement n'a pas été trop important.

Par contre les inconvénients de cette technique, utilisation d'un moniteur de détection de changement et nombre important de paramètres à régler, la rendent difficile à mettre en œuvre et peu flexible.

3.6.2.4 Thermodynamical Genetic Algorithm (TDGA)

En 1996 Mori propose une technique appelée algorithme génétique thermodynamique [Mori 1996]. Cette approche est couplée avec une mémoire qui enregistre les meilleurs individus trouvés dans le passé. Le TDGA s'inspire du principe d'énergie minimale utilisé dans le recuit simulé. L'idée est de sélectionner les individus pour la génération suivante de telle manière que l'énergie F soit minimisée :

$$F = \langle E \rangle - HT \qquad 29)$$

avec $\langle E \rangle$ l'énergie moyenne du système, H l'entropie et T la température.

La diversité de la population est contrôlée en ajustant T dans le temps comme dans le recuit simulé.

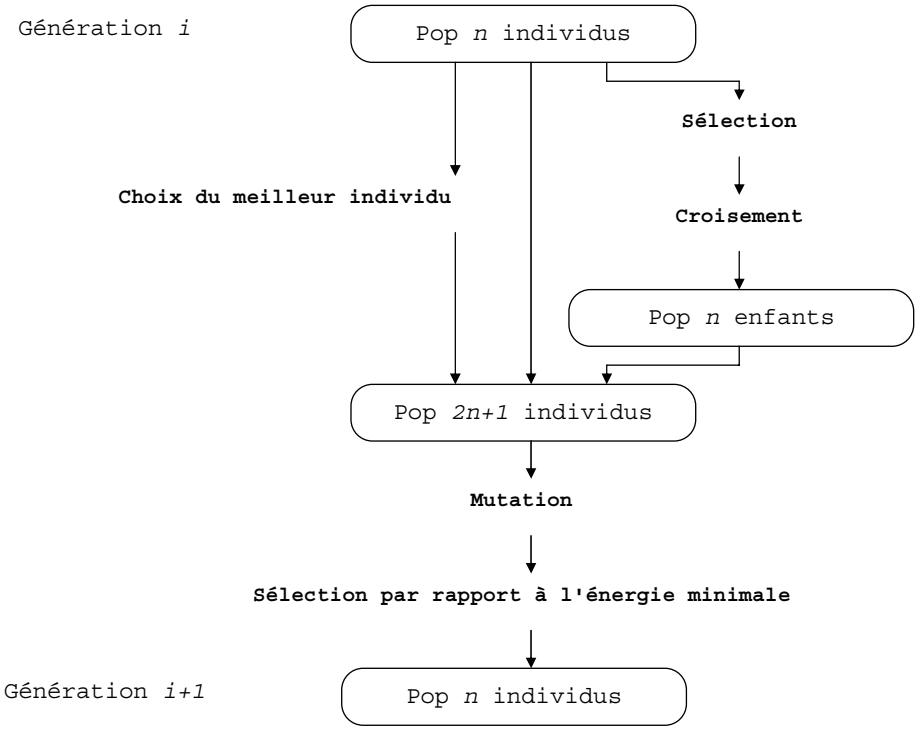


Figure 26 : Schéma de fonctionnement de TDGA

3.6.3 Discussion

L'utilisation de mémoire implicite ou explicite n'est efficace que si l'évolution de l'optimum définit un cycle de temps ou repasse à proximité d'une ancienne solution. Dans le cas contraire, ces méthodes échouent systématiquement.

Ces méthodes sont en général très complexes à mettre en œuvre. Les approches implicites demandent la création délicate d'un génotype et les approches explicites basées sur l'expérience exigent de se poser les questions suivantes : Quelle taille de la population externe ? Quelle stratégie de mise à jour ? Quelle périodicité de mise à jour ? Quelle stratégie de remplacement ? Quel taux de remplacement ?

Enfin, si le changement de l'environnement est trop rapide par rapport à la diffusion des bons schémas, dans le génome des individus pour les approches implicites et dans la population externe pour les approches explicites, ces méthodes vont échouer.

Un moyen de réduire les difficultés liées à ces méthodes est de recourir à des méthodes utilisant plusieurs populations.

3.7 Approche par multipopulation

3.7.1 Heuristique

Ces approches emploient plusieurs populations de manière à répartir les individus sur plusieurs sommets. Ainsi en suivant plusieurs optima différents, la probabilité de suivre l'optimum global est renforcée.

3.7.2 Multinational evolutionary algorithm

Cet algorithme évolutionnaire proposé par Ursem [Ursem 2000] recherche non seulement les solutions globales mais aussi les solutions locales d'un problème. Pour cela, cette méthode prend en compte la topologie de l'espace de recherche pour rassembler les individus en sous-groupes, chacun couvrant une partie de l'espace.

Cette méthode est basée sur l'analogie avec les sociétés humaines et introduit les notions de gouvernement et de nation[51]. La fonction de notation, appelée *Hill-valley*, est utilisée pour détecter les vallées dans l'espace de recherche, faire migrer les individus et décider de regrouper deux nations en une seule.

Une nation regroupe l'ensemble des individus autour d'un même pic et possède un gouvernement et un ensemble de règles d'évolution.

Le gouvernement de chaque nation est constitué des meilleurs individus de celle-ci. Il permet de calculer la valeur approchée du sommet en effectuant la moyenne des positions des individus le constituant.

L'ensemble des règles d'évolution détermine le comportement de tous les individus qui appartiennent à la nation. Les règles établies par Ursem sont :

- la migration des individus et la création d'une nouvelle nation,
- le regroupement de deux nations lorsqu'elles sont situées sur le même sommet,
- la sélection des individus,

- l'élection du gouvernement,
- le croisement restreint,
- l'initialisation d'une nouvelle nation.

Pour rechercher les vallées situées entre deux individus i_p et i_q, Ursem utilise l'algorithme suivant :

Hill-valley (i_p, i_q, *exple*[])

a) min_{fit} = min(fitness(i_p), fitness(i_q))

b) **Pour** *j* **de** *1* **à** *taille_exple* **faire**

Calcul des points $i_{intérieur}$ sur le segment entre i_p et i_q

Si (min_{fit} > fitness($i_{intérieur}$))

alors retourne (min_{fit} - fitness($i_{intérieur}$))

Fin Pour

c) Retourne *0*

Le paramètre *exple* est un tableau de valeurs comprises *entre 0 et 1* qui permet de calculer les positions des points intérieurs $i_{intérieur}$. Dans le cas d'un espace à deux dimensions :

$$i_{intérieur} = (i_{px} + (i_{qx} - i_{px}).exple[j],\ i_{py} + (i_{qy} - i_{py}).exple[j])$$

Si la fonction retourne *0*, cela signifie que l'algorithme n'a pas trouvé de point intérieur donc qu'il n'y a pas de vallée entre les points i_p et i_q.

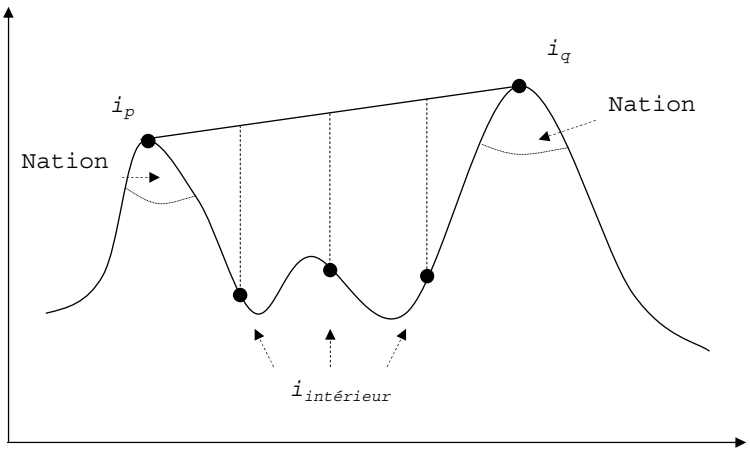


Figure 27 : Algorithme de Hill-valley

[51] Notion similaire à celle de niche écologique.

Critique

Le point positif de cette méthode est son approche discriminante. En effet, lors d'un changement, seules les nations proches de lui subiront une modification, le reste de la population ne changera pas. Cela permet de réduire le temps de calcul et de ne pas détruire les nations situées dans d'autres régions de l'espace.

Même si l'approche de Ursem permet de s'abstraire des heuristiques de partage ou de remplacement pour répartir les individus dans l'espace, elle ne répond pas au problème du rapport entre nombre d'individus de la population et nombre de pics de la fonction.

L'algorithmique de cette méthode apparaît également compliquée et beaucoup de paramètres de réglage sont exigés. Le paramètre le plus discutable est *exple*[], car le choix des différentes valeurs de ce tableau pour discrétiser la recherche des points intérieurs ne peut pas assurer que la réponse de la fonction *Hill-valley* reflètera la bonne topologie de l'espace. On retrouve ici les inconvénients des techniques basées sur un maillage de l'espace de recherche.

3.8 Conclusion sur les méthodes d'optimisation en environnement dynamique

Les méthodes d'optimisation en environnement dynamique se heurtent aux problèmes suivants :

- détecter le changement d'environnement,
- fournir une réponse appropriée à ce changement.

Dans les méthodes présentées ci-dessus, la détection du changement se fait soit de manière endogène soit de manière exogène. La qualité de réponse des méthodes qui utilisent un mécanisme externe de détection de changement, est totalement dépendante de celui-ci. De plus une seule méthode propose une approche discriminante de réponse au changement d'environnement. Mais cette méthode utilise un grand nombre de paramètres de réglage qui la rend peu flexible à mettre en œuvre.

Méthode	Action	Type	Détection du changement	Approche discriminante
Hypermutation	Augmentation de la mutation	Réactive	Exogène	Non
VLS	Augmentation de la mutation	Réactive	Exogène	Non
Random immigrants	Injection de nouveaux individus	Préventive	Endogène	Non
Sharing	Modification de la fitness en fonction du voisinage	Préventive	Endogène	Non
Crowding	Sélection en fonction de la similarité des individus	Préventive	Endogène	Non
Age des individus	Modification de la fitness en fonction de l'âge des individus	Préventive	Endogène	Non
Diploïdie	Modification de la structure des gènes	Mémoire implicite	Endogène	Non
Gène multiniveau	Modification de la structure des gènes	Mémoire implicite	Endogène	Non
Expérience	Injection d'anciens individus	Mémoire explicite	Exogène	Non
TDGA	Sélection en fonction d'une énergie minimale + archive	Mémoire explicite	Endogène	Non
Multinational	Utilisation de plusieurs populations	Multipopulation	Endogène	Oui

Tableau 6 : Récapitulatif des méthodes d'optimisation en environnement dynamique

Dans la méthode que nous proposons (Partie II), nous utilisons un système multiagent. Chaque agent est doté d'un comportement de recherche locale qui lui permet de progresser vers un optimum, et d'un comportement de développement qui lui permet de défendre l'optimum trouvé et de créer d'autres agents. Ces comportements procurent à l'agent une capacité endogène de détection d'un changement dans voisinage et une capacité de réponse rapide et ciblée.

Partie II.

La méthode des gouttes d'eau (water drops)

Chapitre 4.
Méthode de recherche des optima locaux d'une fonction multimodale en environnement dynamique

Nous présentons dans ce chapitre la méthode que nous avons développée. Nous expliquons dans un premier temps les raisons pour lesquelles nous avons choisi une approche locale et multiagent. Ensuite nous décrivons le fonctionnement de notre méthode et nous présentons des résultats obtenus.

Nous avons appelé cette méthode, **les gouttes d'eau** par analogie au phénomène météorologique. Lorsqu'il pleut, une goutte d'eau tombe sur le sol puis ruisselle jusqu'à ce qu'elle s'arrête au fond d'un trou. Ensuite, l'arrivée d'autres gouttes d'eau dans ce trou vient faire grossir la première goutte pour former une flaque. Au bout d'un certain temps, la flaque remplit le trou et elle déborde en envoyant une partie de ses gouttes d'eau remplir le trou voisin. Et ainsi de suite…

Ce phénomène schématise très bien notre méthode. Un agent (la goutte) glisse vers un optimum (le trou) puis il augmente sa zone d'influence (la flaque) et génère d'autres agents pour explorer les optima voisins (le débordement).

4.1 Pourquoi une approche locale et multiagent ?

Comme nous l'avons vu dans le chapitre précédent, une méthode d'optimisation de fonctions multimodales en environnement dynamique doit avoir les qualités suivantes :

- maintenir la diversité dans l'espace de recherche,
- favoriser la création de niches,
- ne pas subir la tromperie de la fonction,

- détecter de manière endogène le changement,
- avoir une approche discriminante qui permette une réponse ciblée.

Dans le cas de fonctions multimodales, les méthodes basées sur les algorithmes génétiques exigent l'utilisation d'une heuristique de répartition et un nombre important d'individus pour pouvoir trouver tous les optima locaux. Les temps de calcul sont importants et la qualité des solutions n'est pas satisfaisante. En effet, ces méthodes ne peuvent pas garantir que tous les optima seront trouvés car elles oublient les petites niches qui ne possèdent pas un pouvoir d'attraction suffisant.

De plus, si un décideur veut utiliser les résultats obtenus, ces méthodes ne peuvent pas lui fournir une estimation de l'erreur faite sur la position de l'optimum trouvé. Donc le risque pris par le décideur ne peut pas être évalué.

Dans le cas d'un environnement dynamique, la plupart des méthodes présentées dans le chapitre précédent ont une approche non discriminante. Dès lors, le temps de réponse au changement de l'environnement et le risque de détruire des niches situées dans d'autres régions de l'espace sont augmentés.

Nous avons développé une méthode multiagent qui utilise une approche de recherche locale inspirée des stratégies d'évolution.

L'approche multiagent doit engendrer une forte capacité d'exploration de tout l'espace de recherche ainsi qu'une capacité discriminante de l'espace.

L'approche locale doit engendrer une forte capacité d'exploitation et d'exploration locales.

Les apports originaux de cette méthode sont l'introduction de **la notion de zone d'influence** d'un agent et **l'utilisation d'un facteur de précision** pour rechercher les optima.

Lors de la conception de cette méthode, nous nous sommes attachés à **réduire le nombre de paramètres de réglage et à rendre ces paramètres intelligibles pour l'utilisateur**.

4.2 Description de la méthode

Le principe de la méthode est basé sur la création aléatoire d'agents logiciels indépendants qui vont ensuite essayer de coloniser un optimum de la fonction. Chaque agent possède une stratégie de recherche locale qui lui permet de trouver l'optimum local le plus influent.

Le système est décomposé en deux parties :

- le système de contrôle,
- les agents.

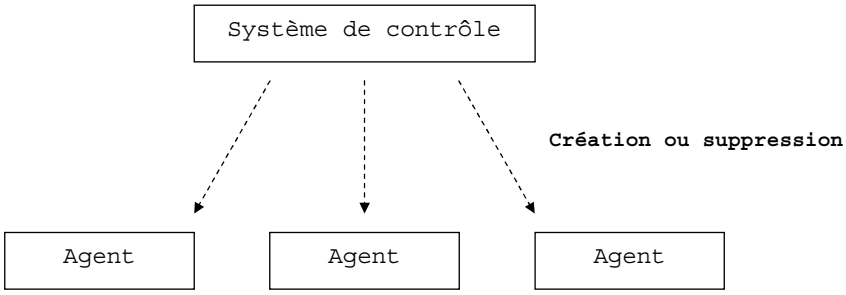


Figure 28 : Schéma du système

4.2.1 Paramétrage de la méthode

Cette méthode est très simple à paramétrer. Après avoir défini la fonction à optimiser ainsi que les dimensions de l'espace de recherche, l'utilisateur doit spécifier deux paramètres qui vont influencer la capacité de recherche et la précision de calcul des solutions trouvées : *force* et *epsilon*.

Le paramètre **force** permet de déterminer la proportion du nombre de points que l'agent va tester à chaque étape de son évolution. Plus ce paramètre est élevé, plus le calcul est lent mais la recherche de l'agent est plus efficace.

Le paramètre **epsilon** détermine la précision souhaitée par l'utilisateur. Chaque agent utilise ce paramètre pour arrêter sa recherche de l'optimum et pour gérer les calculs de ses zones de recherche et d'influence. **Pour un décideur ce paramètre représente le taux d'erreur acceptable**.

4.2.2 Description du système de contrôle

Le système de contrôle a deux rôles différents :

- il crée périodiquement des agents qui sont positionnés aléatoirement dans l'espace de recherche,
- il élimine les agents inefficaces.

La génération aléatoire d'agents apporte à la méthode une bonne capacité d'exploration de l'espace de recherche car chaque point de l'espace a la même probabilité d'être atteint.

L'élimination des agents inefficaces permet de réduire le nombre d'agents présents dans l'espace de recherche. Comme nous le verrons par la suite, cette suppression permet d'accélérer le processus de recherche sans diminuer sa capacité de prospection.

Algorithme du système de contrôle

Notre système de contrôle ne possède pas de critère d'arrêt, il effectue perpétuellement le cycle d'exécution suivant :

a) Création aléatoire d'un ou plusieurs agent(s).

b) Evolution de chaque agent.

c) Elimination des agents inefficaces.

d) Retourner en a).

Plusieurs manières pour fixer le nombre d'agents à ajouter dans l'espace de recherche à chaque début de cycle ont été envisagées :

- ajout d'un seul agent,
- ajout d'un nombre *nb* d'agents.

La première solution est très simple à implanter et n'exige aucun réglage de paramètre.

La deuxième solution a comme avantage d'accélérer le processus de recherche au lancement de l'algorithme. En contrepartie, lorsque tous les optima de la fonction seront trouvés, l'ajout d'un trop grand nombre d'agents augmentera inutilement le temps de calcul d'un cycle d'exécution sans apporter un gain d'efficacité significatif à la recherche. Enfin, cette solution impose à l'utilisateur de choisir le nombre *nb* d'agents créés à chaque cycle. L'efficacité de ce nombre étant dépendante du nombre d'optima de la fonction, il est difficile de justifier son choix lorsque l'environnement est dynamique car il n'existe aucune hypothèse de maintien du nombre d'optima de la fonction.

Finalement, nous avons choisi de privilégier la première solution car elle exempte l'utilisateur du choix d'un ou plusieurs paramètre(s).

4.2.3 Description d'un agent

4.2.3.1 Propriétés

Chaque agent du système a les propriétés suivantes :

- Il a une position $P(x_1, \ldots, x_n)$ dans l'espace de recherche.
- Il développe autour de lui une zone de recherche variable dans laquelle il effectue sa recherche de l'optimum.
- Il est mu par un but à atteindre (par exemple : trouver le maximum ou le minimum d'une fonction).

- Après avoir colonisé un optimum, il tente de le protéger en calculant une zone d'influence autour de cet optimum. Ainsi tout agent pénétrant dans cette zone sera éliminé par le système de contrôle.
- Il est doué d'autonomie, c'est-à-dire qu'il n'est pas dirigé par l'utilisateur ou par le système de contrôle. Il possède une fonction d'évolution qui lui permet de modifier sa position dans l'espace de recherche en fonction du but à atteindre.
- Il ne possède qu'une représentation partielle de son environnement. Il ne connaît pas la fonction qu'il va explorer mais il connaît les domaines de valeurs des variables constituant l'espace de recherche.
- Il peut se reproduire. Dans notre système reproduction signifie duplication. Dans certaines conditions l'agent génère un double. Ainsi ce dernier ira coloniser un optimum voisin.
- Il ne communique pas avec les autres agents mais il perçoit leur présence.
- Il surveille constamment sa zone d'influence. Si un changement d'environnement intervient, provoquant la disparition de cet optimum, l'agent reprend immédiatement sa recherche.

4.2.3.2 Cycle de vie d'un agent

Au cours de sa vie, un agent a deux états possibles. Il peut être **non valide** ou **valide**.

Définition d'un agent valide

On dit qu'un agent est valide lorsqu'il a trouvé un optimum[52]. Par opposition, un agent est dit non valide s'il n'est pas situé sur un optimum.

Pour déterminer s'il est situé sur un optimum local, l'agent va tester la valeur de la fonction pour des points de son voisinage situés à une distance *epsilon*[53]. Si aucun de ces points n'est meilleur que la position de l'agent alors on peut considérer que l'agent est sur un optimum local.

[52] Local ou global. Aucune différence ne peut être faite à ce niveau là.
[53] Paramètre défini dans le paragraphe 4.2.1

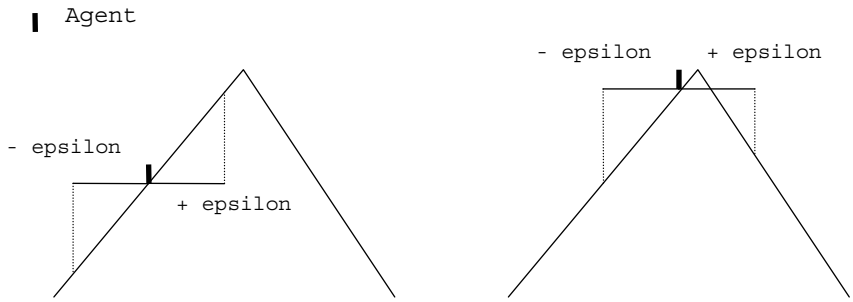


Figure 29 : Définition d'un agent valide dans un problème à une variable

Le schéma ci-dessus montre la différence entre un agent valide (à droite) et un agent non valide (à gauche) pour une fonction $f : R \rightarrow R$. Nous pouvons remarquer sur le schéma de droite que l'agent n'est pas exactement sur l'optimum. Il est tout de même valide car l'erreur, différence entre sa position et la position de l'optimum, est inférieure à *epsilon*.

L'état de l'agent détermine son comportement d'évolution. Si l'agent est non valide, il adopte un comportement de recherche d'optimum (stratégie de recherche). Dans le cas contraire, il adopte un comportement de développement qui comprend l'augmentation de sa zone d'influence (stratégie de défense) et la création d'autres agents (stratégie de diffusion).

A sa naissance un agent est non valide. Il adopte donc une stratégie de recherche de l'optimum.

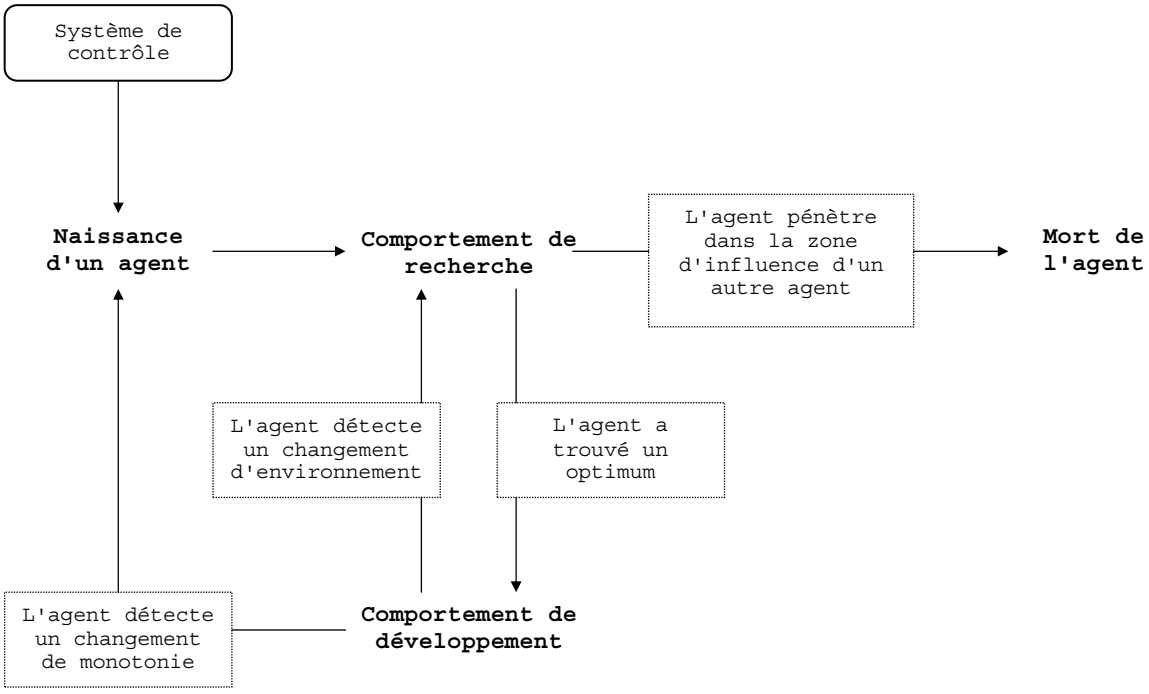


Figure 30 : Cycle de vie de l'agent

Nous avons choisi d'appeler nos différents algorithmes "des stratégies" en raison des similitudes entre les stratégies d'évolution adaptatives et notre méthode. En effet nos agents testent un ou plusieurs points de leur voisinage, choisissent la position du meilleur pour se déplacer et adaptent automatiquement leurs paramètres à la topologie de la fonction.

Stratégie de recherche de l'optimum

Cette stratégie est basée sur l'utilisation d'une zone de recherche *ZR* qui est agrandie ou réduite en fonction de la monotonie de la fonction objectif dans cette zone. *ZR* est définie par la position de l'agent et par une valeur de voisinage $interv_{rech}$ similaire à la valeur σ_{share} utilisée dans l'heuristique de partage.

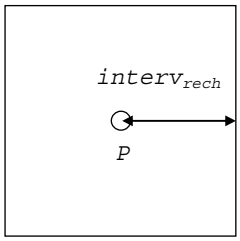


Figure 31 : Zone de recherche dans un problème à 2 variables

Les avantages de cette stratégie de recherche sont :

- l'utilisateur n'a pas besoin de définir un paramètre de voisinage,
- le paramètre de voisinage $interv_{rech}$ s'adapte automatiquement à la topologie locale de la fonction, chaque agent effectuant ainsi une recherche locale indépendamment des autres agents.

A chaque cycle d'évolution, l'agent recherche une zone de monotonie *ZM*, incluse dans *ZR*, autour de lui. Puis il recherche dans *ZM* une position meilleure que sa position courante *P* et se place sur cette position si elle existe.

Si l'agent perçoit que sa zone de recherche *ZR* est monotone alors il l'agrandit, sinon il réduit *ZR* à la plus grande zone de monotonie trouvée. Ce mécanisme permet à l'agent de régler automatiquement sa vitesse de progression vers un optimum en fonction de la topologie de l'espace. Les figures ci-dessous décrivent le mouvement d'un agent ainsi que l'évolution de sa zone de recherche. Du schéma 1 au schéma 2 de la figure ci-dessous, l'agent se déplace et agrandit sa zone de recherche. Du schéma 2 au schéma 3, l'agent se déplace et réduit sa zone de recherche car il a détecté un changement de monotonie de la courbe.

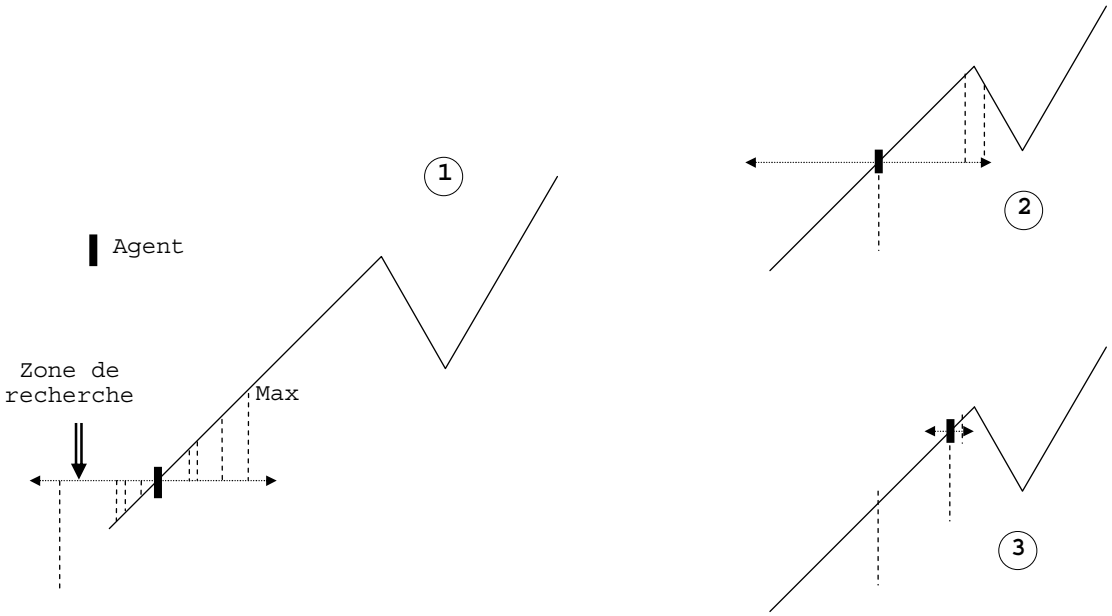


Figure 32 : Mouvement d'un agent

Quand l'agent a trouvé l'optimum, il devient valide et adopte un comportement de développement :

- il acquiert les capacités de développer une zone d'influence et de créer d'autres agents,
- il modifie également sa façon d'estimer la monotonie de la surface,
- il ne modifie plus sa position.

Avant de décrire le comportement de développement (stratégies de diffusion et de défense) de l'agent nous allons décrire la façon dont l'agent analyse la monotonie de sa zone de recherche.

Algorithme de recherche d'une zone de monotonie

A l'aide des schémas n°1 et 2 de la figure ci-dessus nous pouvons voir comment l'agent évalue la monotonie d'une fonction à une variable. Les points choisis aléatoirement dans son voisinage sont classés par abscisses croissantes. Ensuite l'agent teste le sens de variation de la fonction entre sa position et le point d'abscisse la plus proche et il réitère ce test de variation pour chaque couple de points d'abscisses consécutives. Si le sens de variation n'est pas modifié alors la fonction est monotone sur cette zone sinon l'agent enregistre le point au delà duquel la fonction n'est plus monotone. Cette information lui permet de modifier la dimension de sa zone de recherche.

Dans le cas d'une fonction à deux variables, une grille de points choisis aléatoirement peut être envisagée pour analyser la monotonie de la fonction.

Pour les fonctions à n variables il est préférable d'utiliser une technique de type "lancer de rayons". L'agent choisit un nombre *force* de valeurs v_i comprises entre *0* et *1*. Ces valeurs v_i sont triées par

ordre croissant de façon que $v_i \leq v_{i+1}$. Ensuite l'agent[54] choisit aléatoirement un point $(y_0, y_1, \ldots, y_n)$ situé à la limite de sa zone de recherche. Pour terminer il teste l'évolution du sens de variation de la fonction pour la suite de points z_i de coordonnées $(v_i.(y_0-x_0)+x_0, v_i.(y_1-x_1)+x_1, \ldots, v_i.(y_n-x_n)+x_n)$ avec i $\in$ [*1*..*force*]. Au total l'agent lance *n.force* rayons pour tester la monotonie de la fonction dans sa zone de recherche.

L'avantage d'utiliser ce type de technique est de diminuer le coût calculatoire de la méthode de recherche de monotonie. En effet si nous avions gardé une analyse de la monotonie basée sur la création d'une grille de points, ce coût aurait été de $force^n$. En utilisant cette technique nous avons un coût calculatoire de $n.force^2$. De plus l'implémentation de cette technique est très simple à réaliser.

Stratégie de diffusion

Lorsqu'un agent valide détecte un changement de monotonie dans sa zone de recherche, ne pouvant occuper deux optima à la fois, il crée un autre agent qui va aller coloniser l'optimum situé au-delà de ce changement de monotonie. Cette capacité permet au processus de recherche de se diffuser localement dans l'espace. Ce mécanisme est schématisé ci-après (Figure 33).

Stratégie de défense de l'optimum

Cette stratégie est basée sur le développement d'une zone d'influence *ZI* autour de la position de l'agent (Figure 33). Cela va permettre de supprimer tous les agents qui essayeront de coloniser ce sommet.

La zone d'influence d'un agent valide est l'espace qui l'entoure dans lequel l'agent ne détecte aucun changement de monotonie de la fonction objectif.

Cette stratégie peut être qualifiée de stratégie de formation de niche. Car, en agrandissant sa zone d'influence, l'agent définit un voisinage autour de l'optimum similaire à l'espace qu'occuperait une niche composée de plusieurs individus.

Les avantages de cette stratégie sont :

- un seul individu est nécessaire pour maintenir un optimum local,
- pas d'utilisation de technique de sharing donc aucun réglage de paramètre,
- la taille de la zone d'influence s'adapte à la topologie de l'espace autour de l'agent.

[54] La position de l'agent est notée : $x_0, x_1, \ldots, x_n$.

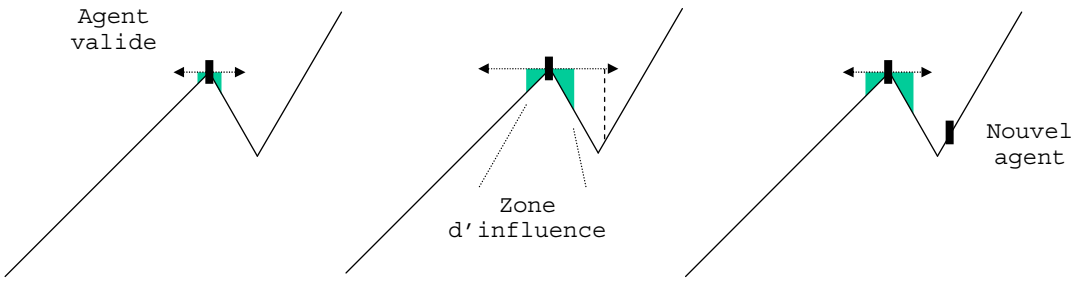


Figure 33 : Développement de la zone d'influence

4.2.3.3 Algorithme de fonctionnement d'un agent

Lors de la création d'un agent, sa position $P(x_1, \ldots, x_n)$ est initialisée aléatoirement et sa zone de recherche *ZR* est définie par les intervalles $[x_1\text{-}epsilon, x_1+epsilon]$, ... , $[x_n\text{–}epsilon, x_n+epsilon]$ et il n'a aucune zone d'influence *ZI*.

Par la suite chaque agent a le même fonctionnement :

a) Tirage aléatoire d'un ensemble de points dans la zone de recherche ZR de l'agent. Le nombre de points dépend d'une valeur, appelée *force*, attribuée aux agents.

b) Calcul de la valeur de la fonction à optimiser en chaque point.

c) Recherche d'une zone de monotonie *ZM* autour de sa position *P*.

d) Recherche de la position P_{max} du maximum dans la zone de monotonie *ZM*.

e) **Si** P_{max} est différent de *P* **alors** l'agent change de position $P \leftarrow P_{max}$

f) **Si** il y a monotonie sur l'ensemble de la grille de points

alors extension de la zone de recherche *ZR*

sinon réduction de la zone de recherche $ZR \leftarrow ZM$

h) **Si** l'agent est valide

alors /*----- Agent valide -----*/

Si il y a monotonie

alors agrandissement de la zone d'influence *ZI*

sinon l'agent donne naissance à un nouvel agent

sinon /*----- Agent non valide -----*/

sa zone d'influence disparaît, $ZI \leftarrow \varnothing$

4.3 Caractéristiques de la méthode

Compromis entre exploitation et exploration

L'utilisation combinée des agents et du système de contrôle donne à cette méthode une bonne capacité d'exploration ainsi qu'une bonne capacité d'exploitation.

La capacité d'exploitation de cette méthode est due aux caractéristiques de l'agent. La stratégie de recherche utilisée permet aux agents de progresser rapidement vers les optima.

La capacité d'exploration de la méthode est un compromis entre l'exploration globale de l'espace d'état par le système de contrôle et l'exploration locale des agents. En développant une zone d'influence autour d'eux pour défendre leur optimum, les agents valides créent un phénomène de répulsion qui disperse les autres agents dans l'espace de recherche.

Diffusion dans l'espace de recherche

Lorsqu'un agent a trouvé un optimum, il acquiert la capacité de création. Les nouveaux agents vont ainsi explorer le voisinage à la recherche d'autres optima. La répétition de ce phénomène va permettre de visiter tout l'espace de recherche.

Exploration des espaces non connexes

Pour accélérer l'exploration de l'espace de recherche, le système de contrôle crée périodiquement des agents. Ainsi tous les points de l'espace ont la même probabilité d'être atteints. Un autre avantage de ce processus est de permettre la recherche dans des espaces non connexes. Il suffit alors qu'un seul agent soit positionné dans une des parties de l'espace pour que celle-ci soit totalement visitée par diffusion.

Dynamique du système

A tout moment l'environnement de l'agent peut être modifié. Les agents subissant ce changement se redistribuent très vite dans l'espace de recherche afin de résoudre le nouveau problème. Comme chaque agent surveille sa zone d'influence, il peut percevoir le changement d'environnement et à partir de sa position il va recommencer sa recherche d'un optimum.

Le schéma ci-dessous illustre le changement de position d'un optimum.

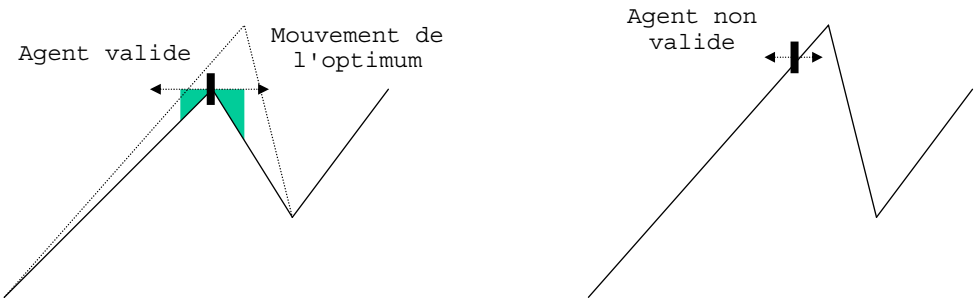


Figure 34 : Changement d'optimum

Dans l'exemple de dynamique de la figure ci-dessus, la position de l'optimum est modifiée. L'agent précédemment positionné sur cet optimum détecte ce changement et modifie son état. Il devient non valide. A partir du cycle suivant, il adopte une stratégie de recherche de l'optimum.

Le schéma ci-dessous montre l'apparition d'un optimum dans le voisinage d'un agent valide.

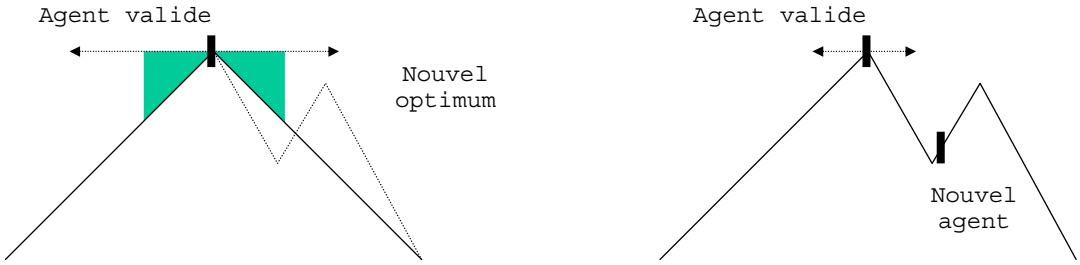


Figure 35 : Apparition d'un nouvel optimum

Dans l'exemple de la figure ci-dessus, l'apparition du nouvel optimum provoque un changement de monotonie de la courbe. L'agent détecte ce changement de monotonie à l'intérieur de sa zone d'influence. Il réduit alors sa zone d'influence et sa zone de recherche puis il crée un nouvel agent dans la zone de changement de monotonie.

Auto-régulation du nombre d'agents

L'utilisation de la zone d'influence des agents valides permet d'éliminer les agents qui progressent vers un optimum déjà colonisé. Ainsi le système de contrôle régule lui-même le nombre d'agents dans l'espace de recherche. Plus le nombre d'agents valides augmente, plus le total des zones sous influence augmente. Donc l'espace restant à explorer diminue. Une autre conséquence bénéfique est la diminution du nombre d'agents présents dans l'espace de recherche. Cette caractéristique est montrée par le résultat obtenu (Figure 39) pour l'optimisation d'une fonction fortement multimodale (Figure 38).

Une approche discriminante

Cette caractéristique est la qualité principale de cette méthode. En effet, vous avons vu dans les chapitres précédents que les méthodes qui n'ont pas une approche discriminante ne peuvent pas

donner une réponse appropriée au changement. De plus, ces méthodes risquent de détruire des niches situées dans d'autres régions de l'espace.

Notre méthode, par l'intermédiaire des agents, permet d'apporter une réponse précise et rapide au changement d'environnement (Figure 34 et Figure 35). Car seul(s) le ou (les)[55] agent(s) situé(s) dans la zone de changement sera(seront) impliqué(s). Tous les autres agents n'auront même pas connaissance de ce changement et ils n'en subiront aucune conséquence.

Détection endogène

Notre méthode n'a pas besoin d'un mécanisme extérieur de détection de changement de l'environnement. Ainsi sa robustesse ne dépend pas de ce mécanisme ou d'un paramétrage que devra effectuer l'utilisateur.

La détection est basée sur une procédure simple de détection de changement de monotonie.

4.4 Résultats

Nous présentons dans cette section les résultats obtenus par cette méthode. Nous montrons tout d'abord les capacités de cette méthode à trouver tous les optima locaux d'une fonction dans des problèmes multimodaux et à réguler automatiquement le nombre d'agents présents dans l'espace de recherche. Ensuite nous présenterons le test que nous avons mis en place pour tester cette méthode dans un environnement dynamique.

4.4.1 En environnement statique

Les figures ci-dessous montrent une comparaison entre un algorithme génétique (à gauche) utilisant une technique de sharing et notre méthode (à droite). Cette comparaison est destinée à montrer l'efficacité de notre méthode à trouver tous les optima d'une fonction à *epsilon* près. Pour cela nous utilisons deux courbes *f1* et *f2*. *f1* possède 3 optima et *f2* possède 17 optima.

[55] Dans les schémas de la Figure 35, au pire, deux agents devront modifier leur stratégie.

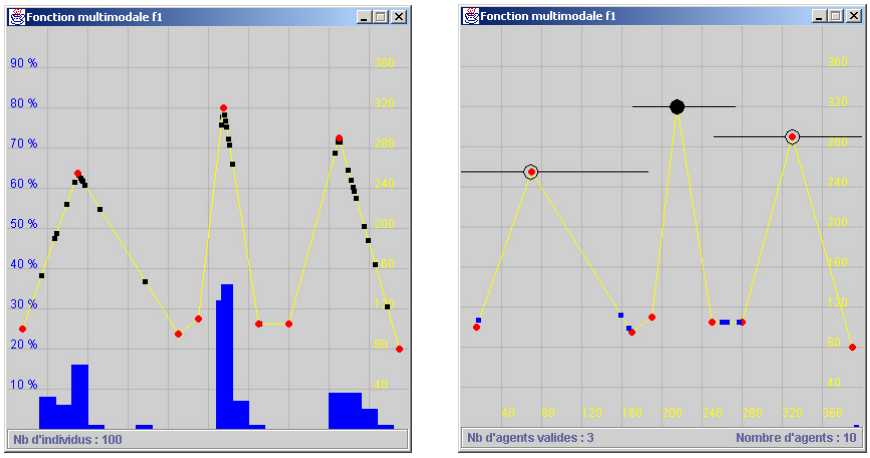


Figure 36 : Fonction *f1*

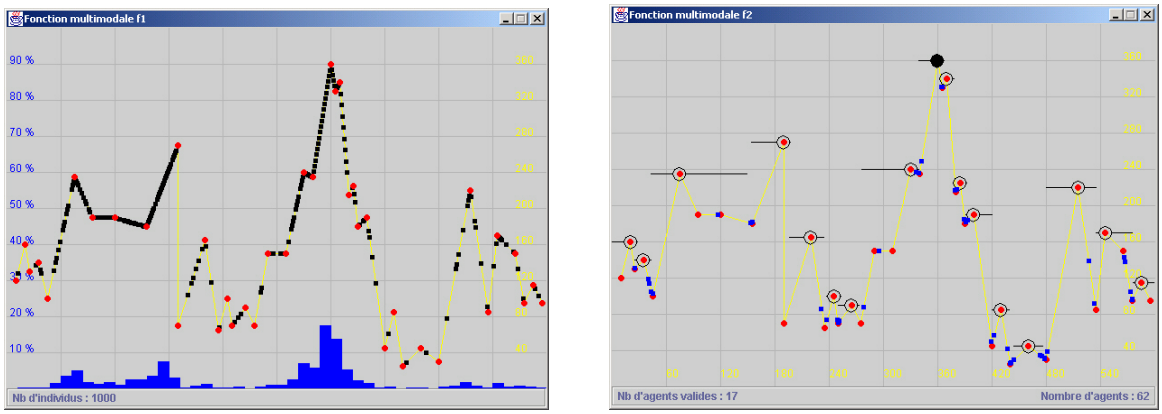


Figure 37 : Fonction *f2*

Nous pouvons constater que l'utilisation d'une technique de sharing classique avec un algorithme génétique permet de distinguer les 3 optima de *f1*. Mais cette technique a du mal à trouver tous les optima de *f2*. Beaucoup de petites niches ne sont pas détectées. Pour *f2*, il faut également remarquer le nombre important d'individus dans la population.

Notre méthode distingue très bien dans les deux cas tous les optima locaux et l'optimum global. Nous pouvons aussi constater le nombre très faible d'individus nécessaires pour explorer l'espace de recherche (10 pour la fonction *f1* et 62 pour la fonction *f2*) après stabilisation du système. C'est-à-dire après que tous les optima ont été trouvés et que chaque agent sur un optimum a développé sa zone d'influence.

Dans l'exemple ci-dessous, la fonction possède 761 optima dont 18 optima globaux. On constate que tous les optima ont été trouvés, et on aperçoit clairement les zones d'influence de chaque agent (Figure 38).

$$f(x,y) = \sum_{i=1}^{5} i*\cos((i+1)*x+i) * \sum_{i=1}^{5} (-i*\sin((i+1)*y+i))*(i+1) \; +$$

$$\sum_{i=1}^{5} i*\cos((i+1)*y+i) * \sum_{i=1}^{5} (-i*\sin((i+1)*x+i))*(i+1)$$

avec $-10 \leq x,y \leq 10$

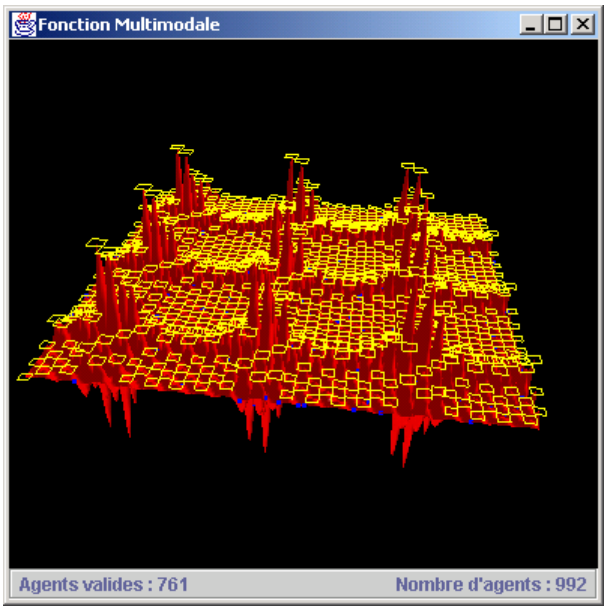


Figure 38 : Fonction à 761 optima dont 18 optima globaux

Le graphique ci-dessous représente l'évolution du nombre d'agents dans l'espace de recherche par rapport au nombre d'agents valides[56] pour la fonction multimodale présentée ci-dessus.

Dans un premier temps, l'évolution est approximativement linéaire car les zones d'influence ne sont pas assez nombreuses pour compenser l'introduction des nouveaux agents par le système de contrôle.

A partir du 250ième optimum trouvé, les zones d'influence compensent de plus en plus l'augmentation du nombre d'agents dans l'espace de recherche. Cela explique le début d'inflexion de la courbe.

A partir du 500ième optimum trouvé, l'inflexion de la courbe s'accentue. Le nombre d'agents dans la population diminue car les zones d'influence permettent de supprimer plus d'agents que le système de contrôle n'en crée.

Une fois que tous les optima sont trouvés et que chaque agent valide a développé sa zone d'influence, le nombre total d'agents se stabilise[57] au alentour de 950. En effet le nombre d'agents

[56] C'est-à-dire par rapport au nombre d'optima trouvés car un agent dit valide est un agent qui a trouvé un optimum.
[57] N'apparaît pas sur ce type de courbe.

créés par le système de contrôle ou par les agents valides qui détectent un changement de monotonie, est proche du nombre d'agents détruits car entrés dans une zone d'influence. La différence entre le nombre total d'agents et le nombre d'agents valides représente les agents en situation de recherche d'optimum.

L'avantage de ce phénomène de baisse du nombre total d'agents est l'augmentation de la vitesse d'exploration de l'espace lorsque les zones d'influence compensent la création d'individus.

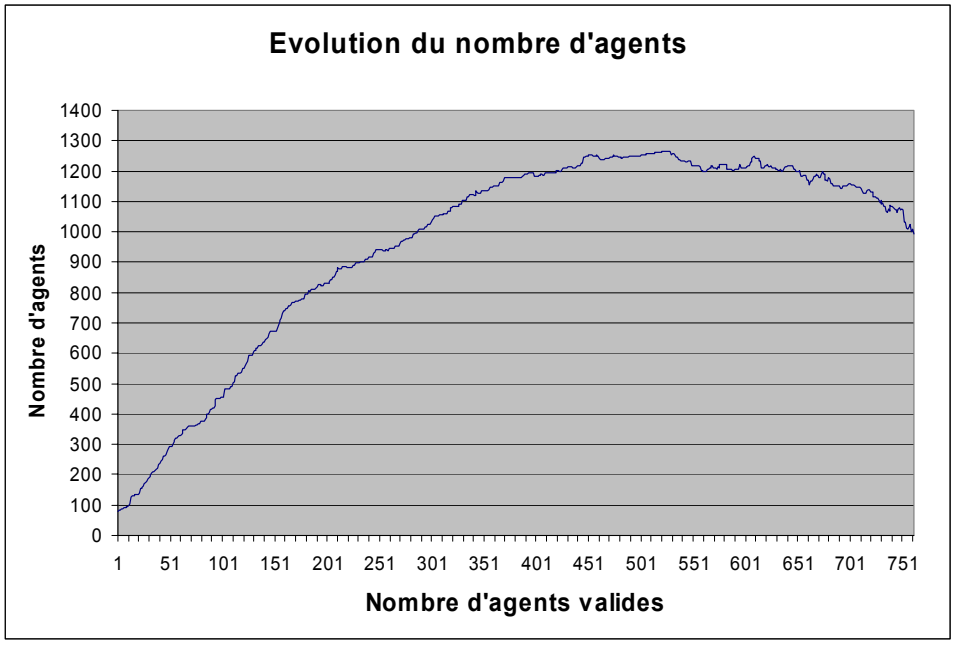


Figure 39 : Courbe d'évolution du nombre total d'agents par rapport au nombre d'agents valides

Par opposition, l'utilisation d'un algorithme génétique dans ce type de problème exige que le nombre d'individus dans la population soit approprié par rapport au nombre d'optima de la fonction. Cela implique que l'utilisateur ait une connaissance a priori du problème.

Lobo et Harik ont essayé de créer une méthode basée sur les algorithmes génétiques qui tente de réguler automatiquement le nombre d'individus [Harik and Lobo 1999]. Dans leur approche, ils démarrent avec une population très faible, puis ils doublent la taille de la population chaque fois que la perte de diversité génétique est trop rapide. Cette méthode demande de nombreux tests avant de trouver la taille adéquate de la population et elle dépend de la mesure de la perte de diversité.

4.4.2 En environnement dynamique

Pour tester notre méthode en environnement dynamique, nous avons créé un test appelé **test de la boule de billard**. La fonction est représentée par une ligne brisée constituée par un ensemble de points (en rouge). Ces derniers sont classés par abscisses croissantes. La ligne brisée est formée par les segments (en jaune) qui joignent deux points consécutifs. Tous les points sont fixes sauf un : la boule de billard.

Au départ, la boule de billard est située en haut à gauche de l'espace de recherche. Elle est donc l'optimum de la fonction. Ensuite elle parcourt le chemin indiqué en pointillé en 2000 pas de temps. Au fur et à mesure de son déplacement la boule de billard modifie la courbe. Les points formant la courbe étant classés par ordre d'abscisse croissante, chaque fois que la boule de billard change de position dans le classement la ligne brisée est modifiée.

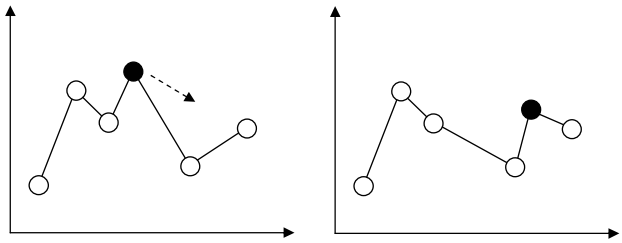


Figure 40 : Influence du déplacement de la boule de billard sur la ligne brisée

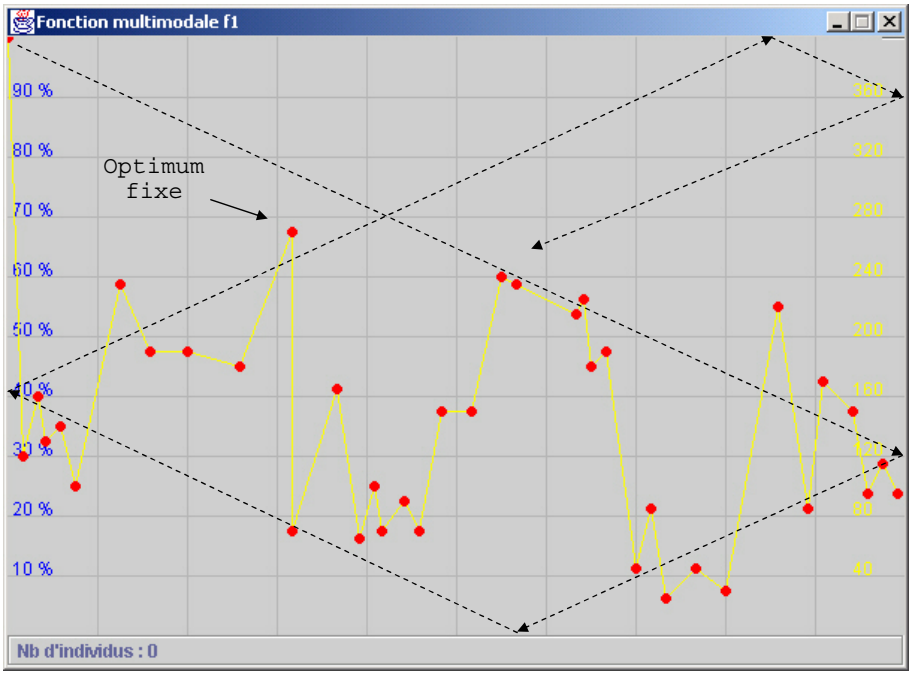


Figure 41 : Le test de la boule de billard

A certaines périodes, la boule de billard modifie également la position de l'optimum global (Figure 42). Deux périodes sont intéressantes à étudier (du temps 1 à 300 et du temps 1400 à 2000). En effet durant ces périodes la position de l'optimum de la fonction est modifiée à chaque cycle d'exécution. En dehors de ces périodes, l'optimum représenté par le point appelé "optimum fixe" ne change pas.

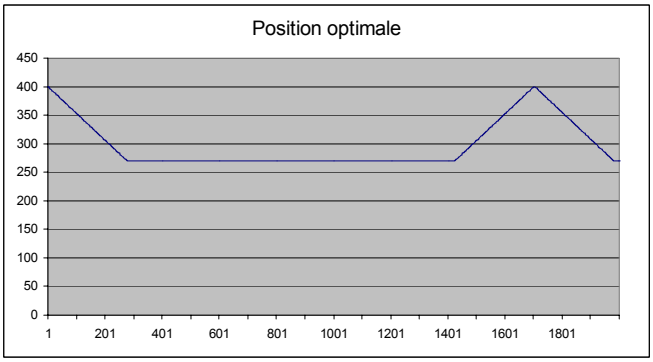


Figure 42 : Evolution de la valeur optimale

Nous avons comparé notre méthode (avec *epsilon = 0.0001* et *force = 5*) à :

- Un algorithme génétique simple doté d'un taux de mutation de 5%. Ce taux élevé permet de simuler l'hypermutation décrite au paragraphe 3.4.2.1. La population est de 100 individus et le taux de croisement est de 60%.
- Un algorithme génétique avec sharing classique qui a une population de 400 individus, un taux mutation de 0,5% et un taux de croisement de 60%.

Pour effectuer les mesures présentées dans les graphiques ci-dessous nous avons synchronisé le mouvement de la boule de billard avec notre méthode et avec les algorithmes génétiques. La boule de billard est déplacée d'une valeur Δx sur l'axe des x et Δy sur l'axe des y, à chaque génération pour les algorithmes génétiques, à chaque cycle d'exécution pour notre méthode.

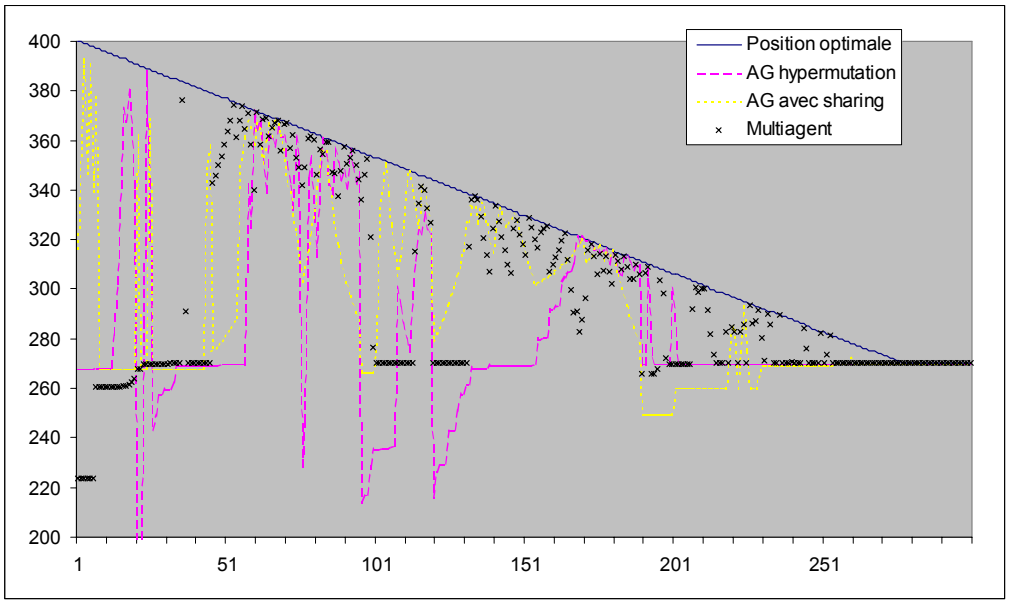


Figure 43 : Graphique sur la période 1

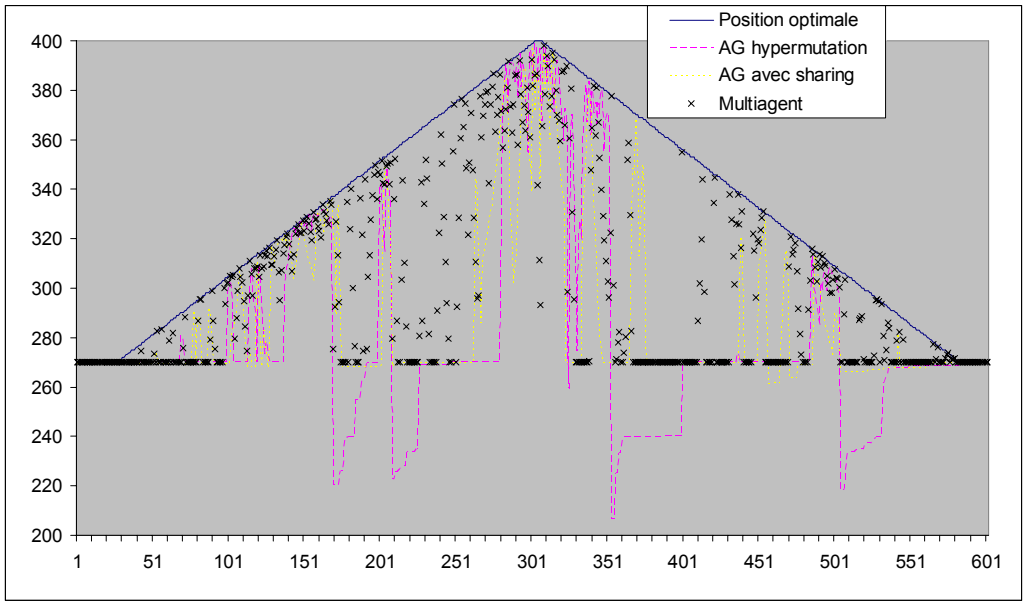


Figure 44 : Graphique sur la période 2

Les graphiques ci-dessus représentent les évolutions des meilleurs individus à chaque cycle d'exécution. Nous pouvons ainsi comparer l'efficacité de notre méthode à celle d'une méthode utilisant l'hypermutation et d'une méthode utilisant le sharing dans un environnement dynamique.

Dans la période 1 (Figure 43), notre méthode est plus proche de l'optimum dans 74% des cycles d'exécution par rapport à l'algorithme génétique avec l'hypermutation, et dans 68% pour l'algorithme avec sharing.

Dans la période 2 (Figure 44), notre méthode est plus proche de l'optimum dans 85% des cycles d'exécution par rapport aux deux algorithmes génétiques. Les résultats sont meilleurs dans la période 2 car, lorsque cette période commence, la plus grande partie de l'espace de recherche est sous influence d'un agent. Donc, la réponse de notre méthode est plus rapide.

Le scénario de la période 1 est le pire des cas pour notre méthode car, dès le départ, l'optimum est en mouvement. A ce moment là, il n'y a qu'un seul agent dans tout l'espace de recherche contre 100 individus pour la méthode utilisant l'hypermutation. Il faut donc un certain laps de temps (environ 50 cycles d'exécution) pour que le nombre d'agents soit suffisant pour découvrir l'optimum global.

On peut également remarquer sur les deux graphiques que notre méthode ne subit pas la tromperie de la courbe multimodale. Par moment, les deux méthodes utilisant un algorithme génétique ont un maximum qui descend sous la valeur 270. Cette valeur est l'optimum fixe (Figure 41) et seule la boule de billard peut avoir une valeur plus grande qu'elle. Donc, si une méthode ne trouve pas la boule de billard, elle devrait se positionner en second lieu sur cet optimum. Or, on constate que la méthode basée sur l'hypermutation est trompée très souvent et ses résultats sont parfois très éloignés de l'optimum. La méthode basée sur le sharing résiste beaucoup mieux à cette tromperie.

Notre méthode n'est jamais trompée car un des agents s'est positionné sur l'optimum fixe et le maintient durant toute la simulation.

4.5 Conclusion

Nous avons réalisé une méthode multiagent qui permet de rechercher tous les optima d'une fonction multimodale à *epsilon* près. Cette méthode est également capable de répondre de manière discriminante au changement d'environnement. Ceci lui procure une capacité à être utilisée dans les problèmes en environnement dynamique. Comme nous l'avons constaté dans les résultats, elle ne subit pas la tromperie de la fonction car elle arrive à positionner un agent sur chaque optimum local. Cette méthode est également très simple à paramétrer pour un utilisateur et ses paramètres sont compréhensibles.

Bien évidemment ces conclusions devront être confirmées par un plus grand nombre de tests. Nous espérons aussi développer un ensemble de problèmes pour tester les méthodes en environnement dynamique car il n'existe actuellement aucun jeu de test validé par la communauté scientifique permettant d'effectuer une comparaison entre les différentes méthodes.

Suite à ces résultats encourageants, nous avons souhaité généraliser cette approche pour résoudre des problèmes multiobjectifs.

Chapitre 5.
Généralisation à des problèmes d'optimisation multiobjectifs

Dans ce chapitre nous présentons la généralisation de notre méthode multiagent à des problèmes d'optimisation multiobjectifs. Ensuite, nous commentons les résultats obtenus sur ce type de problèmes et nous posons la réflexion suivante : la zone d'influence de l'agent peut-elle être interprétée par le décideur comme une mesure de marge de sécurité ?

5.1 Introduction

La méthode présentée dans le chapitre précédent utilise un système multiagent pour rechercher l'ensemble des optima d'une fonction mutimodale. Nous souhaitions conserver les principales qualités de cette approche pour l'optimisation de problèmes multiobjectifs :

- auto-adaptation du nombre d'agents à l'environnement,
- peu de paramètres de réglage,
- réponse discriminante et détection endogène du changement.

Nous utiliserons la dominance[58] au sens de Pareto pour effectuer une comparaison entre les différentes solutions et pour diriger les agents dans l'espace de recherche. Donc, dans les problèmes multiobjectifs, nos agents ne recherchent pas un optimum mais une zone Pareto-optimale, c'est-à-dire une zone dans laquelle tous les points sont non dominés.

Comme pour l'approche précédente le cycle de vie d'un agent est décomposé en deux phases. Une première phase de recherche dans laquelle l'agent cherche une zone Pareto-optimale et une deuxième phase de développement dans laquelle l'agent augmente sa zone d'influence et génère d'autres agents pour explorer le voisinage.

5.2 Les modifications apportées

Plusieurs modifications ont été apportées pour adapter notre méthode :

- la définition de la validité de l'agent,
- la redéfinition de la zone d'influence
- le mouvement de l'agent,
- le rôle du paramètre *epsilon*,
- la détection des autres agents dans le voisinage.

Un agent sera dit **valide** s'il a développé autour de lui une zone d'influence. Dans cette approche, la zone d'influence est appelée **zone de dominance** car elle détermine la région autour de l'agent dans laquelle toutes les solutions potentielles sont non dominées.

Figure 45 : Zone de dominance et de recherche d'un agent

5.2.1 Mouvement de l'agent

Pour modifier sa position dans l'espace de recherche l'agent va analyser la dominance dans son voisinage, c'est-à-dire à l'intérieur de sa zone de recherche, en effectuant la procédure ci-dessous :

[58] Définie au paragraphe 2.7.1.2.

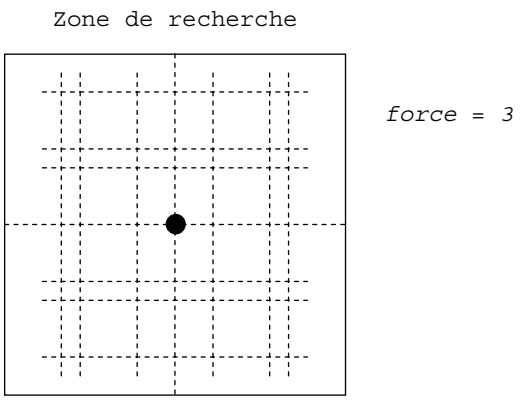


Figure 46 : Grille de dominance dans un exemple à deux fonctions objectifs

a) Découpage de la zone de recherche en une grille de dominance (Figure ci-dessus) de taille dépendante du paramètre *force*.

b) On calcule pour chaque point *i* de la grille le nombre de points *j* qui le dominent. On obtient ainsi une **matrice de dominance**.

c) **Si** l'agent est non valide

alors

Il recherche une zone autour de lui dont les points sont non dominés.

Si cette zone existe

alors l'agent peut développer sa zone de dominance.

sinon l'agent calcule les sommes des lignes et des colonnes de la matrice de dominance, puis il choisit les plus petites valeurs trouvées pour déterminer les coordonnées du point sur lequel il va se placer. Il agrandit également sa zone de recherche.

sinon

Il effectue la somme $somme_{int}$ des valeurs des points de la matrice de dominance situés dans sa zone de dominance, et la somme $somme_{ext}$ des valeurs des points situés hors de la zone de dominance.

Si $somme_{int} = 0$

alors

Il tente d'agrandir sa zone de dominance en effectuant un éventuel déplacement.

Si $somme_{ext}$!= *0*

alors il crée un autre agent dans la zone extérieure à sa zone de dominance. Cet agent va aller explorer le voisinage.

sinon /*----- Cela signifie que l'environnement a changé ----*/

Il recalcule sa zone de dominance et sa zone de recherche.

5.2.2 Le rôle d'*epsilon*

Dans cette approche multiobjectif, *epsilon* joue le rôle d'un critère d'arrêt. Lorsque l'agent trouve une zone de non dominance, il tente à chaque cycle de temps de développer cette zone au maximum pour agrandir son influence. S'il n'arrive pas à agrandir cette zone pendant un certain nombre de cycles de temps max_{cycle}, il stoppe sa phase de développement. Ensuite il se contente de surveiller sa zone de dominance pour détecter un éventuel changement en testant un nombre *force* de points choisis aléatoirement.

Ce nouveau paramètre introduit n'est pas primordial pour notre méthode car il n'a aucun effet sur le comportement d'optimisation de l'agent. Il a été introduit pour éviter qu'un agent essaie d'augmenter sa zone de dominance alors que cette zone a atteint son maximum. Cela permet une économie de calculs.

5.2.3 La détection des autres agents dans le voisinage

Pour inciter les agents valides à se disperser dans l'espace et pour éviter que leurs zones de dominance se recouvrent, un agent *i* ajoute la valeur *1* à tout point de son voisinage inclus dans la zone de dominance d'un autre agent *j* si la taille de la zone de dominance de *j* est supérieure à celle de *i*. Ainsi, l'agent *i* détecte qu'une certaine partie de sa zone de dominance est déjà sous influence d'un autre agent donc *i* devra réduire sa zone de dominance.

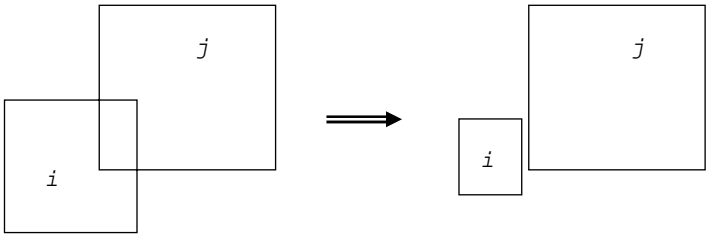


Figure 47 : Détection des autres agents dans le voisinage

Ce mécanisme de répulsion permet de disperser les agents valides sur l'ensemble de la zone Pareto-optimale. Ainsi nous n'avons pas besoin d'utiliser une heuristique de répartition.

5.3 Résultats et réflexion

Nous avons testé notre méthode sur les problèmes de Viennet 1 et 2 (Paragraphe 2.9).

Pour le premier problème, nous avons effectué une comparaison entre avec la méthode NPGA[59] et notre méthode. Les résultats sont montrés ci-dessous, NPGA (Figure de gauche) et notre méthode (Figure de droite). Le triangle présent sur les deux figures représente l'ensemble Pareto-optimal.

[59] Décrite au paragraphe 2.7.2.3.

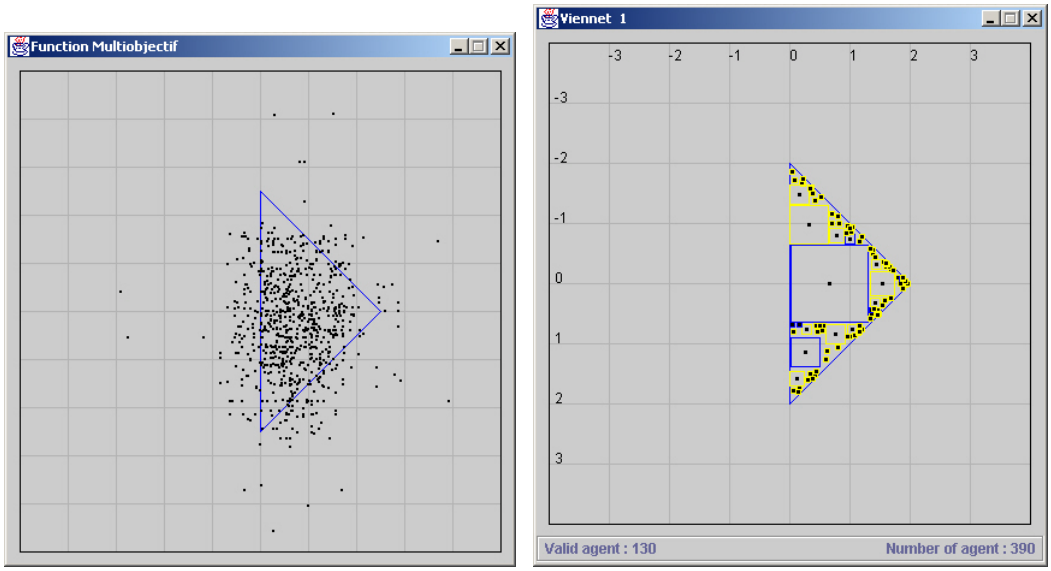


Figure 48 : Résultats pour le problème de Viennet 1

La première caractéristique que l'on remarque est la différence de représentation des solutions obtenues. Dans la méthode NPGA, comme dans toutes les méthodes décrites dans le Chapitre 2, les solutions sont représentées par des points alors que notre méthode présente les solutions sous forme de zones Pareto-optimales. Cette représentation permet une description plus fine de l'ensemble Pareto-optimal du problème et mène à la réflexion suivante :

La zone d'influence, ou zone de dominance, de l'agent peut-elle être interprétée par le décideur comme une mesure d'une marge de sécurité ?

A priori, le décideur ne connaît pas la forme de l'ensemble Pareto-optimal avant de résoudre son problème. Une méthode comme NPGA va proposer un ensemble de solutions Pareto-optimales. Ensuite il va choisir une des solutions en fonction de ses préférences sur les résultats obtenus. Envisageons le cas présenté dans la figure suivante :

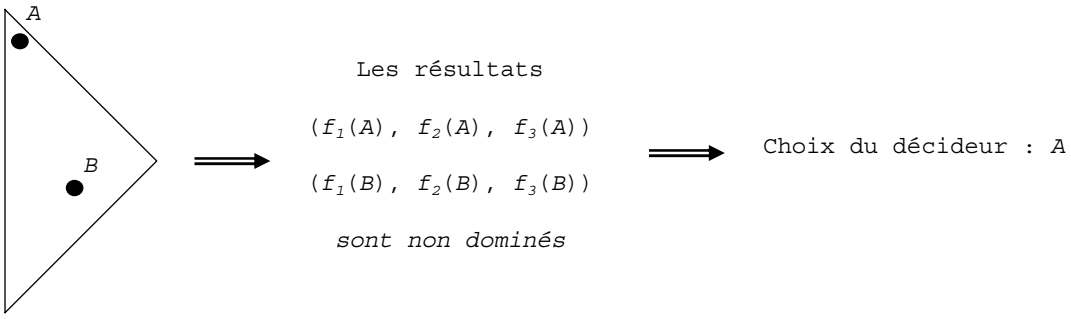


Figure 49 : Choix du décideur

Les deux solutions *A* et *B* offrent des résultats équivalents, le décideur opte pour la solution *A*. Or on constate sur la figure que *A* est relativement proche de la zone non optimale alors que *B* est plus éloigné de cette zone. Donc en choisissant *A* le décideur prend plus de risque car, si pour une raison

quelconque une des composantes de l'action A doit être légèrement modifiée, A deviendra non Pareto-optimale.

L'utilisation d'une méthode basée sur un algorithme génétique n'offre aucune information supplémentaire pour permettre une meilleure prise de décision alors que notre méthode propose en plus de la solution Pareto-optimale une mesure permettant d'estimer la marge de sécurité du décideur. Cette mesure est la largeur de la zone de dominance de l'agent. Par exemple, dans le résultat de notre méthode Figure 48, l'agent situé au centre du schéma possède une grande zone de dominance par rapport aux autres agents. Donc en choisissant la solution proposée par cet agent, le décideur se donne une marge de sécurité par rapport à l'action à effectuer.

Nous avons également testé notre méthode sur le problème n°2 de Viennet. Dans ce cas, l'ensemble des solutions Pareto-optimales est non convexe.

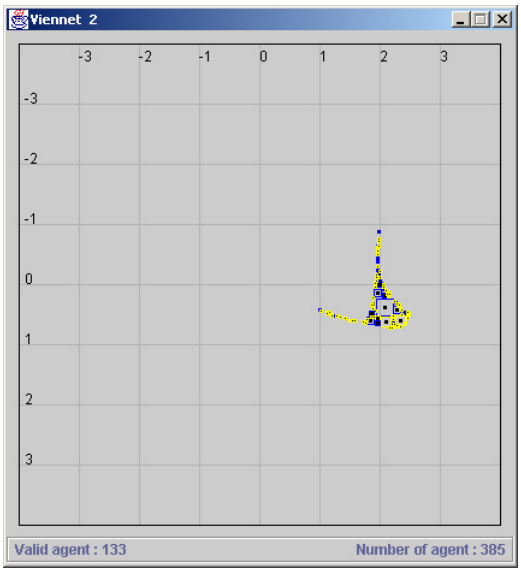


Figure 50 : Résultat pour le problème de Viennet 2

On peut constater que notre méthode arrive également à définir un ensemble Pareto-optimal non convexe.

5.4 Conclusion

La généralisation de notre approche à des problèmes multiobjectifs n'a pas nécessité l'abandon de ses caractéristiques principales. Notre méthode est simple à paramétrer et le nombre d'agents s'adapte automatiquement au problème à optimiser. Par contre nous ne sommes pas capables, faute de tests sur des problèmes dynamiques et multiobjectifs, de confirmer ses capacités de détection endogène du changement et de réponse discriminante.

Nous devons également tester notre approche sur un plus large éventail de problèmes multiobjectifs, notamment sur des problèmes dont la frontière de Pareto est non convexe. Les problèmes 1 et 2 exposés par Viennet ont une frontière de Pareto convexe, donc toute solution Pareto-optimale locale est une solution Pareto-optimale globale. Si un problème a une frontière de Pareto non convexe alors il apparaît une distinction entre les solutions Pareto-optimales locales et les solutions Pareto-optimales globales (schéma ci-dessous). Or l'état actuel d'avancement de notre méthode ne lui permet pas de distinguer ces deux types de solutions.

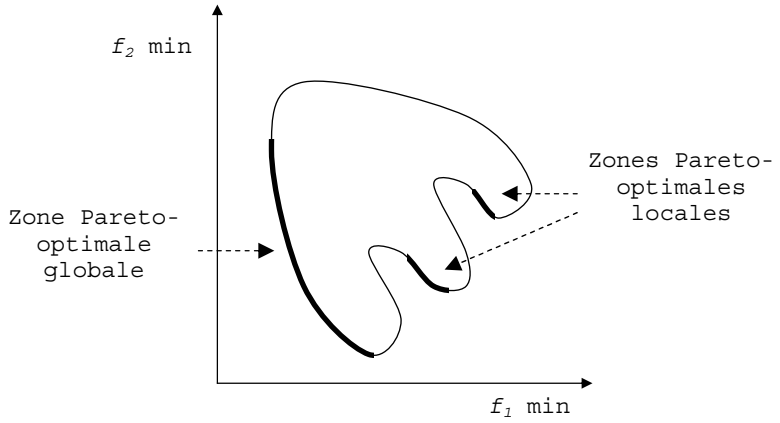


Figure 51 : Zones Pareto-optimales locale et globale

L'apport intéressant de notre méthode multiobjectif est cette présentation différente des solutions Pareto-optimales. Au lieu de donner au décideur un ensemble de solutions Pareto-optimale, notre approche lui indique un ensemble de zones Pareto-optimales. Cette représentation lui apporte ainsi une meilleure connaissance du problème.

Nous avons enfin émis une réflexion sur l'interprétation possible de la taille de la zone de dominance de nos agents comme l'évaluation d'une marge de sécurité pour le décideur.

Partie III.

Simulations économiques

Nous présentons dans cette partie deux applications de méthodes évolutionnaires dans le cadre de problèmes économiques. Le lecteur sera sûrement étonné de constater que la méthode des gouttes d'eau que nous développons dans la partie précédente de cette thèse n'est pas utilisée dans ces simulations. La première application qui sera présentée dans cette partie constitue le point de départ de cette thèse. Il s'agissait de réaliser une plate-forme générique pour la simulation de coopération et de compétition entre acteurs économiques. Les recherches et les réflexions menées pour réaliser les modèles de simulation ont progressivement orienté nos travaux théoriques vers les problèmes d'optimisation multiobjectifs et dynamiques. Ensuite, nous n'avons pas eu le temps de mettre en œuvre cette méthode dans le cadre de problèmes économiques.

Dans une première partie nous présentons une plate-forme générique pour la simulation de coopération et de compétition entre agents économiques. Cette partie se termine par un exemple d'application effectué en collaboration avec un étudiant en DEA 2IL, Stéphane Sanchez. Cette application utilise un algorithme génétique pour optimiser le placement des succursales d'une entreprise en fonction de la répartition géographique des consommateurs.

Dans une deuxième partie nous présentons un travail effectué en collaboration avec Isabelle Leroux, étudiante préparant une thèse en économie au sein du LEREPS. Cette simulation porte sur l'évolution des comportements de négociations. Dans ce cas l'algorithme génétique n'est pas utilisé comme une méthode d'optimisation mais comme un mécanisme évolutionnaire. En effet le but de cette étude n'est pas d'optimiser une fonction mais d'étudier l'évolution des proportions des différents types d'individus présents dans la population. Nous verrons comment cet algorithme nous permet d'obtenir des équilibres entre différents groupes d'individus constituant la population.

Chapitre 6.
Une plate-forme générique pour la simulation de coopération et de compétition entre agents économiques

6.1 Introduction

Peu de modèles actuels de simulation économique prennent en compte à la fois les notions de temps et d'espace géographique dans lequel les acteurs sont répartis. Ce chapitre discute de la mise en place d'une plate-forme de simulation économique visant à maintenir la réelle complexité du système[60]. L'objectif est de réaliser un ensemble d'outils génériques qui permettent, à un économiste ou un informaticien, de créer et de faire interagir les différents agents économiques : les entreprises, les ménages, les institutions financières et l'état. Cette plate-forme doit permettre également d'ajouter des facteurs exogènes aux agents, c'est-à-dire des facteurs qui échappent à tout contrôle des agents (conditions climatiques, sources de matières premières, etc.). Dans une première partie, nous présentons les différents modèles nécessaires à une simulation réaliste d'une société économique. Ensuite nous présentons le moteur d'algorithme génétique que nous avons conçu afin de résoudre des problèmes d'optimisation uniobjectifs ou multiobjectifs. Enfin nous présenterons une première utilisation du moteur développé : une entreprise proposant plusieurs services essaie d'optimiser le placement de ses succursales dans un espace géographique dans lequel les consommateurs sont répartis de façon non homogène.

[60] Pour comprendre un système compliqué, il faut le simplifier, pour comprendre un système complexe on doit le modéliser car la simplification amène à détruire son intelligibilité, Jean Louis Le Moigne.

6.2 La modélisation d'un système économique

6.2.1 La modélisation des paramètres exogènes

Ces paramètres peuvent être classés en deux types différents. Le premier type regroupe les données variables qui changent de façon autonome et inexorable (exemple : le climat), le second regroupe des données constantes qui ne peuvent pas varier sans une intervention extérieure (exemples : ressources minières, qualité des sols). Nous considèrerons dans un premier temps que les acteurs économiques ne peuvent pas affecter la valeur des paramètres exogènes (exemple : comportement entraînant une pollution de l'environnement) ou prédire leur évolution. Ils devront simplement intégrer ces paramètres comme des contraintes. Par la suite, nous introduirons des comportements d'acteurs altérant les paramètres exogènes (exemple : pollution industrielle, traitement chimique des sols) et des comportements prévoyant l'évolution de certains facteurs exogènes (exemple : météo, étude des sols).

Nous allons discuter maintenant de la façon de modéliser nos différents acteurs économiques.

6.2.2 Les modèles de comportement des acteurs économiques

Chaque acteur économique cherche à optimiser une fonction d'adaptation qui lui est propre. Par exemple, un consommateur peut rechercher à habiter dans un lieu proche de son travail et d'une zone commerciale pour avoir accès à un maximum de commerces, il peut aussi chercher simplement à accumuler le plus d'argent possible en négligeant sa qualité de vie. Une entreprise va vouloir optimiser son processus de production, optimiser l'emplacement de ses points de vente ou rechercher à offrir le dividende le plus important à ses actionnaires. Ces exemples montrent la diversité de comportements que l'on peut rencontrer dans un monde économique. C'est l'une des raisons pour lesquelles nous avons choisi de modéliser les acteurs économiques à l'aide de techniques évolutionnaires. L'application de ces procédés à la simulation d'acteurs économiques permet d'envisager de nouvelles formes de modélisation de comportements collectifs et individuels dans des environnements dynamiques.

Dans un premier temps nos travaux portent essentiellement sur la modélisation des entreprises et des ménages afin de pouvoir mettre en oeuvre très rapidement des simulations de marchés virtuels de consommation. Les modélisations des institutions financières et de l'état ont été simplifiées car ces derniers ne jouent pas, pour le moment, un rôle très important dans nos simulations.

6.2.2.1 L'état

Le rôle de l'état est d'imposer des règles communes à tous les acteurs ou à une certaine catégorie d'acteurs. Ces règles sont vues par les agents économiques comme des contraintes environnementales dont ils doivent tenir compte dans leur recherche d'un comportement adapté.

6.2.2.2 Les consommateurs

Cette catégorie fait l'objet d'une attention particulière. Dans de nombreux modèles économiques, les consommateurs sont remplacés par une fonction de consommation agrégée qui est un paramètre exogène aux entreprises. Cette simplification pose deux problèmes principaux :

- Elle ne prend pas en compte la notion de répartition géographique des consommateurs. Ainsi on considère que tous les consommateurs sont des clients potentiels d'une entreprise. Dans ce cas, les entreprises adoptent principalement des comportements "gloutons" car elles souhaitent satisfaire la demande la plus grande. L'introduction d'un espace géographique dans lequel les consommateurs seront répartis de manière non homogène va faire apparaître naturellement la notion de proximité de consommation.
- Cette fonction de consommation est souvent indépendante du temps. Les entreprises se trouvent alors face à une demande qui n'évolue jamais. Le moment de la transaction entre le consommateur et l'entreprise est aussi totalement occulté, excluant ainsi la notion de consommation saisonnière qui entraîne, de par sa nature, un comportement stratégique spécifique des entreprises.

Pour créer un environnement dynamique dans lequel la consommation évolue dans le temps mais aussi dans l'espace, le consommateur sera modélisé par un ensemble de caractéristiques propres (âge, revenu, nombre d'enfants, etc.) et par un module comportemental simple qui lui permettra de faire un classement des produits qu'il désire acquérir en fonction de ses goûts et de ses besoins. L'évolution du goût du consommateur s'effectuera soit de façon aléatoire, soit par imitation de ses voisins, soit en recherchant le produit qui satisfait au mieux sa demande.

6.2.2.3 Les entreprises

On définit une entreprise comme **une unité de décision économique qui utilise du travail et du capital pour produire et vendre des biens ou des services dans un but de pérennité et de profit**.

Parmi les modélisations d'agents économiques qui doivent être effectuées, celle du comportement d'une entreprise est la plus délicate. En ce qui concerne le consommateur, nous avons considéré que celui-ci évoluait indépendamment des autres consommateurs, or le comportement d'une entreprise doit tenir compte non seulement de son environnement mais aussi du comportement des autres entreprises. Sa pérennité dépend non seulement de son système de production mais aussi des décisions prises en matière de production, de recherche et développement et de coopération/concurrence.

Notre modèle d'entreprise sera constitué :

- d'un système de décision représentant l'équipe dirigeante de l'entreprise,

- d'un système de recherche et développement,
- d'un système de production.

Le système de décision qui représente l'équipe dirigeante pourra être modélisé par un système de classifieurs. Son but est de diriger les systèmes de production et de recherche et de développement en fixant les objectifs à atteindre. Dans le cas où le système de production peut influencer plusieurs variables de sortie, le système de décision pourra lui imposer d'optimiser une variable particulière.

Le système de recherche et de développement permettra de modéliser l'évolution de la qualité de l'entreprise : a) recherche d'un nouveau processus de production permettant d'augmenter la rentabilité de l'entreprise ou b) recherche d'une meilleure qualité des produits proposés au consommateur. Ces deux processus ne seront utilisés que lorsque les changements de qualité de l'entreprise sont essentiels dans sa course au profit ou à sa pérennité. En fonction du type de recherche et développement utilisé par l'entreprise, celle-ci s'engagera soit dans une concurrence par les prix (a), soit dans une concurrence par la qualité (b).

Le système de production sera modélisé par un algorithme génétique. Son but sera d'optimiser une ou plusieurs variables en sortie (outputs) en fonction de variables en entrée (inputs) et des contraintes environnementales.

Le paragraphe suivant présente le moteur générique d'algorithme génétique que nous avons réalisé.

6.3 Le moteur générique d'algorithme génétique

Notre première implémentation de la plate-forme a consisté à créer un moteur générique pour l'optimisation par algorithme génétique. Ce moteur a été programmé en Java et offre la possibilité, de par sa souplesse de conception, d'implémenter facilement des problèmes différents. Le but est d'offrir à l'utilisateur une librairie d'algorithmes génétiques lui permettant d'implémenter facilement son problème sans avoir à se soucier du fonctionnement de l'algorithme.

La classe **Gene** est une classe polymorphe. L'utilisateur peut en fonction de son problème créer ses propres gènes. Il suffit pour cela qu'il définisse les fonctions d'initialisation, de mélange et de mutation de son gène. Ensuite celui-ci pourra être utilisé comme les gènes de base (bit, entier et réel) déjà inclus dans la plate-forme. L'utilisateur peut aussi créer un gène complexe en combinant plusieurs gènes de base.

La classe **Chromosome** est décomposée en deux sous classes : les chromosomes de taille fixe et les chromosomes de taille variable. Chacune de ces classes offre à l'utilisateur plusieurs modes de croisement et de mutation paramétrés à la création du chromosome mais pouvant être changés durant le processus d'évolution.

La classe **Génome** regroupe en fonction du problème un ou plusieurs chromosomes. L'utilisateur peut ainsi créer un génotype ayant ses caractéristiques positionnées sur plusieurs chromosomes différents. Nous avons fait ce choix car lors de la création du génotype d'un individu, tous les caractères génétiques ne sont pas forcément de même nature. L'utilisateur va donc coder son génotype avec des types de données différents et il peut souhaiter également séparer ses caractéristiques afin d'éviter de créer une dépendance qui n'existe pas dans la réalité.

La classe **Population** regroupe les caractéristiques de tous les individus et les paramètres de l'évolution (taux de mutation, taux de croisement, nombre d'individus, mode de sélection, mode de mutation) ainsi que des fonctions statistiques pour les contrôler.

La classe **Evolution** est une tâche Java. L'utilisateur doit simplement spécifier la condition d'arrêt, s'il y en a une, de l'évolution. Les tâches Java permettent de réaliser facilement des simulations dans lesquelles peuvent évoluer plusieurs agents avec un comportement différent. Ce style de conception est très utile pour créer des environnements dans lesquels des agents évoluent indépendamment de leurs voisins. Dans les systèmes ainsi créés, les agents participent à une coévolution par l'intermédiaire d'une mise en concurrence autour du partage d'une ou de plusieurs ressources communes. Une interface permettant de modifier les paramètres de l'algorithme génétique pendant l'évolution a aussi été créée.

Cette librairie a été réalisée avec le souci de simplifier l'implémentation d'un problème d'optimisation basé sur un algorithme génétique. L'utilisateur doit :

- définir son génotype,
- définir sa fonction de notation,
- choisir les paramètres d'évolution.

Le paragraphe suivant présente la première réalisation issue de cette librairie. Nous souhaitions réaliser un outil générique qui permette à une entreprise de calculer un placement optimal de ses points de vente ou de ses points de production dans un environnement complexe contenant des consommateurs et des concurrents.

6.4 Optimisation par algorithme génétique du placement des succursales d'une entreprise

La première application est l'optimisation des placements des succursales d'une entreprise afin d'obtenir le recouvrement le plus efficient possible sur une population de consommateurs. Un recouvrement est dit efficient si un nombre minimal de succursales permet de prendre en charge un

nombre maximal de consommateurs. L'entreprise dispose de différents types de succursales de capacités différentes qui proposent des services complémentaires[61] ou concurrents[62].

6.4.1 Définition du génotype

Les consommateurs

Chaque consommateur est défini par des critères physiques et sociaux. Les critères physiques sont : la position dans l'espace, l'âge et le sexe. Les critères sociaux sont : le nombre d'enfants et le statut (salarié, étudiant, marié…). L'ensemble de ces critères permet de sélectionner un segment de consommateurs afin de le mettre en relation avec le service correspondant proposé par l'entreprise.

La répartition des consommateurs dans l'espace géographique est faite de façon à simuler une situation réaliste. Des zones denses représentent les villes et les critères sociaux sont répartis dans toute la population.

Pour cette première étude les consommateurs n'ont pas de comportement évolutif, il n'y aura donc pas de déplacement de population ni de vieillissement ou de changement de statut social.

L'entreprise

L'entreprise est représentée par l'ensemble de ses succursales. Chacune d'entre elles est définie par sa position (x, y), son rayon d'influence r et sa capacité de traitement cap. Le rayon d'influence permet de simuler l'attraction exercée par une succursale sur le segment de consommateurs visé. Ce critère symbolise le type ou le nombre de services que propose une succursale. La capacité de traitement représente le nombre de consommateurs que peut prendre en charge correctement une succursale. Ce critère symbolise la taille d'une succursale et permettra d'évaluer l'efficacité de celle-ci. Si sa capacité est trop faible, une succursale ne pourra pas traiter correctement tous ses consommateurs potentiels situés dans son rayon d'influence et, au contraire, si elle est trop importante une partie de sa capacité de traitement sera inexploitée.

Les caractéristiques d'une succursale sont modélisées par un gène complexe :

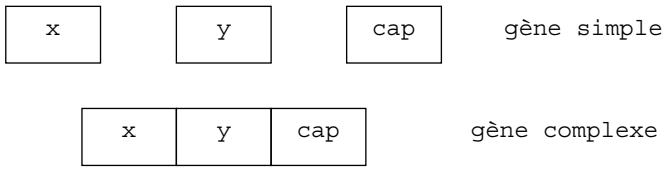


Figure 52 : Le gène représentant une succursale

L'ensemble des succursales d'une entreprise est représenté par un chromosome composé de gènes définis précédemment.

[61] Chaque service dépend d'un type de succursales et il est adapté à un segment particulier de la population.

Figure 53 : Chromosome représentant une entreprise

L'optimisation est effectuée par une seule entreprise. Mais il est possible d'intégrer au modèle : la concurrence déjà en place, les équipements de transports ou tout autre élément pouvant influencer l'optimisation.

6.4.2 La fitness

La résolution du problème se base uniquement sur la qualité de recouvrement de l'ensemble des succursales sans tenir compte du coût de leur implantation. Cependant la prise en compte de ces critères est facilement intégrable au modèle en modifiant la fonction d'évaluation.

Le calcul de la note finale du génome est basé essentiellement sur le compte du nombre de consommateurs recouverts par l'ensemble des succursales constituant le chromosome. Cependant, considérer un tel compte comme seul élément d'évaluation ne permet pas d'obtenir un résultat optimal. En effet, cela conduit à une solution où tout l'espace de travail est recouvert de façon aléatoire sans aucun souci d'optimisation. Donc, pour que l'algorithme converge vers un résultat optimal, c'est-à-dire avec un recouvrement intersuccursales quasi nul et une bonne adaptation des capacités de traitement de chaque succursale, il a fallu introduire des variables supplémentaires pour l'évaluation de la note finale du génome.

$NCgene_i$: nombre de consommateurs traités par la succursale *i* du chromosome évalué. Un consommateur n'est jamais compté deux fois au cours de l'évaluation. Il appartient à une et une seule succursale.

$CAPgene_i$: capacité de traitement d'une succursale *i*.

$EFFgene_i$: efficacité d'une succursale *i* du chromosome évalué. Cette variable symbolise l'adaptation de la capacité de traitement d'une succursale. Elle doit être inférieure ou égale à un.

$$EFFgene_i = \begin{cases} \dfrac{NCgene_i}{CAPgene_i} & \text{si } NCgene_i \le CAPgene_i \\ \dfrac{CAPgene_i}{NCgene_i} & \text{sinon} \end{cases} \qquad 30)$$

NCseg : nombre total de consommateurs du segment visé par le chromosome évalué.

NCchromo : nombre de consommateurs traités par un ensemble de *n* succursales (chromosome).

[62] Certaines succursales peuvent proposer un ou plusieurs services identiques. Donc elles se concurrencent sur une partie

La fonction de notation choisie est la suivante :

$$f(t) = \frac{NCchromo}{NCseg} * 100 - \sum_{i=1}^{TailleChomo} (1 - EFFgene_i) \qquad 31)$$

$f(t)$ représente l'efficience du chromosome moins un malus qui dépend de l'efficacité de chaque succursales qui le compose. Cette fonction d'évaluation s'avère efficace, mais elle a cependant une faiblesse. En effet, un consommateur n'est jamais compté deux fois au cours de l'évaluation mais, en raison du malus minoré par l'efficacité des succursales, des phénomènes de recouvrements plus ou moins importants entre les zones d'influence des succursales d'un même chromosome apparaissent. Le calcul de $f(t)$ ne permet pas de savoir si une zone est traitée par plusieurs agences de petite taille ou par une seule agence de capacité plus importante. Afin de minimiser ces recouvrements entre succursales, un traitement préliminaire sur les chromosomes est appliqué :

- on évalue l'efficacité de chaque succursale composant le chromosome,
- les succursales sont comparées entre elles de manière à détecter si elles se recouvrent ou non, un seuil de tolérance du recouvrement étant possible,
- si le recouvrement entre deux succursales est intolérable alors la plus efficace est maintenue et l'autre est éliminée du chromosome,
- le nouveau chromosome est réévalué, en interdisant cette fois qu'un consommateur soit compté plusieurs fois.

6.4.3 Résultats

Dans les résultats présentés, nous travaillons sur une population de 10000 consommateurs répartis dans un espace de 400x400. Des zones plus denses représentant 85% de la population sont créées afin de simuler des pôles d'attraction.

Notre algorithme génétique :

- 500 individus,
- une sélection par tournoi,
- un taux de croisement de 65%,
- un taux de mutation de 5%,
- le meilleur individu est copié dans la génération suivante.

de la population.

Nous avons envisagé deux stratégies de résolution :

- La première utilise un chromosome variable (Figure de gauche ci-dessous) dans laquelle l'algorithme doit trouver le nombre optimal de succursales à placer.
- La seconde utilise un chromosome de longueur fixe (Figure de droite ci-dessous). Dans cette stratégie le chromosome de chaque individu est augmenté de 1 gène lorsque la moyenne devient très proche de la valeur maximale et réduit de 1 gène lorsque la note maximale n'augmente plus.

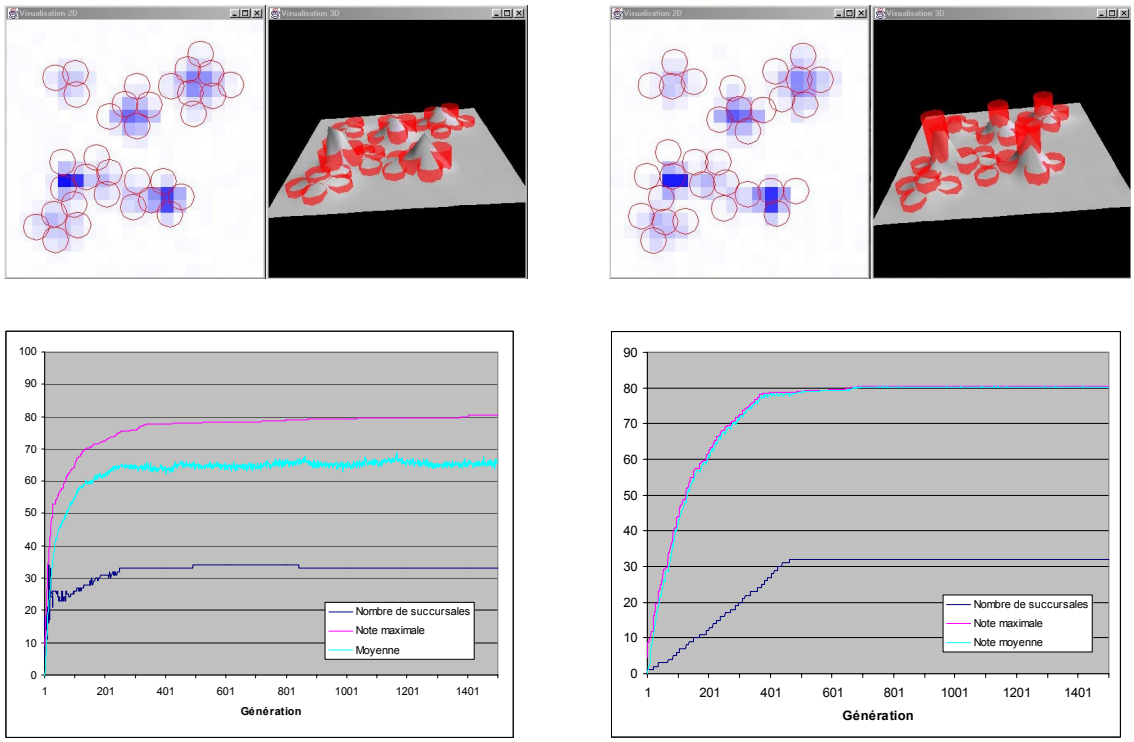


Figure 54 : Résultats des simulations

Ces deux méthodes obtiennent des résultats satisfaisants car elles arrivent à répartir correctement les succursales en fonction de la densité de la population. Pour la stratégie utilisant le chromosome variable, 84% des consommateurs sont traités par 33 succursales. Pour l'autre stratégie, 87% des consommateurs sont traités par 32 succursales. D'un point de vue de temps de calcul, la stratégie utilisant un chromosome fixe est nettement plus rapide.

6.5 Conclusion

Les premières simulations ont donné des résultats encourageants notamment sur la qualité de la solution trouvée par l'algorithme génétique. De plus, la souplesse de conception du moteur d'algorithme génétique nous permet d'ajouter facilement de nouvelles contraintes environnementales ou de modifier les objectifs de l'entreprise.

Actuellement notre moteur d'algorithme génétique a été utilisé pour résoudre des problèmes de placement d'objets dans une scène 3D construite par un modeleur déclaratif et a été inséré dans un système de classifieurs.

Dans nos prochaines simulations nous souhaiterions doter nos consommateurs d'un comportement qui leur permettra de se déplacer dans l'espace géographique. Nous souhaitons ainsi voir émerger de la simulation les phénomènes d'agglomérations locales observés dans la réalité.

Ensuite nous généraliserons le modèle en incluant dans le même environnement plusieurs entreprises concurrentes. Nous souhaitons notamment étudier l'influence de la proximité dans l'émergence de comportements stratégiques locaux.

Chapitre 7.
Simulation de comportements de négociation

7.1 Introduction

La notion de coordination a très souvent été associée dans la littérature économique à celle de coopération, oubliant les conflits d'intérêt et les conflits de pouvoir dont le traitement peut conduire à un résultat coopératif. Or l'enjeu actuel est justement de donner un contenu à cette notion de coordination, en pénétrant les mécanismes qui conduisent effectivement les différents acteurs économiques à se coordonner dans le temps et dans l'espace des activités économiques, au-delà des conflits d'intérêts et de pouvoir. Et pénétrer dans la boîte noire de la coordination revient selon nous à pénétrer dans les mécanismes d'élaboration et de sélection des règles de négociation. Dans cette perspective, l'objet de ce travail est d'offrir un éclairage particulier à la notion de négociation en choisissant comme voie de recherche une approche comportementale basée sur la vie artificielle.

Nous prenons comme point de départ le modèle évolutionniste d'Ellingsen [Ellingsen 1997] qui pose les bases d'une approche de la négociation en termes d'asymétrie communicationnelle. Cet auteur développe en effet une réflexion sur les comportements à la fois adaptatifs et communicationnels d'un jeu de demande de Nash sous ultimatum. Il montre ainsi la convergence systématique vers des stratégies d'appropriation dites justes, où chacun des agents demande 50% du gâteau plutôt que d'adopter des stratégies sophistiquées coûteuses.

Cependant, au-delà du modèle original d'Ellingsen, il s'agit ici de construire les bases d'une analyse évolutionniste des comportements de négociation exprimés non seulement en termes de partage, mais aussi en termes de pouvoir. Les simulations, basées sur un algorithme génétique, feront ainsi apparaître qu'une négociation n'est pas toujours l'expression d'un altruisme partagé, mais au contraire un équilibre dissymétrique entre domination et concession. Nous mettrons en évidence

l'existence du pouvoir du faible de Schelling [Shelling 1960] en situation d'asymétrie communicationnelle.

7.2 Présentation du modèle d'Ellingsen

L'intérêt majeur du modèle d'Ellingsen est d'identifier les raisons qui permettent d'expliquer pourquoi, dans un jeu de négociation bilatérale non anonyme à *n* joueurs, les négociateurs insistent obstinément pour avoir 50% du gâteau plutôt que d'adopter des stratégies sophistiquées coûteuses. Le jeu de négociation porte sur le partage d'un gâteau dans un jeu de demande symétrique sous ultimatum. Il fait interagir des **agents obstinés**, dont les demandes sont indépendantes de celles des adversaires, et des **agents sophistiqués** qui adaptent leur demande à la demande espérée des adversaires plutôt que de repartir les mains vides.

7.2.1 Les bases du modèle d'Ellingsen

7.2.1.1 La fonction de paiement

Le jeu se déroule sur la base d'une population *P* composée de *n* agents dont les stratégies de demande sont hétérogènes. A chaque période, tous les agents de la population sont regroupés aléatoirement par paires, et chaque paire négocie le partage d'un gâteau de taille égale à *1*. Pour cela les agents effectuent **une demande de part de gâteau** comprise dans l'intervalle [*0*, *1*]. Si le cumul des deux demandes est supérieur à la taille du gâteau alors il y a échec de la négociation et le gain de chaque agent est nul. Dans le cas où la demande cumulée est inférieure ou égale à la taille du gâteau alors il y a réussite de la négociation, et les agents se partagent le gâteau proportionnellement à leur demande. La fonction des gains est la suivante :

$$\Pi_{ij} = \begin{cases} \dfrac{i}{i+j} & \text{si } i+j \le 1 \\ 0 & \text{sinon} \end{cases} \qquad 32)$$

Remarque : Dans le cas où la demande cumulée est inférieure strictement à la taille du gâteau ($i+j<1$) alors il existe un **surplus** ($1-i-j$). Ce dernier est partagé proportionnellement à la demande de chaque agent.

7.2.1.2 Les stratégies

Le modèle d'Ellingsen présente deux types de stratégies :

- La stratégie **obstinée**. L'agent adoptant cette stratégie n'est pas capable d'identifier le type (obstiné ou sophistiqué) de son adversaire, et sa demande de part de gâteau est fixée indépendamment de la demande de l'adversaire (stratégie aveugle).

- La stratégie **sophistiquée** ou **responsable**. L'agent adoptant cette stratégie est capable d'identifier le type de son adversaire. Il adapte donc sa demande à celle de son adversaire de manière à éviter l'échec de la négociation.

La matrice des demandes peut être présentée de la manière suivante :

	Obstiné	Sophistiqué
Obstiné	*i* / *j*	*1-j* / *j*
Sophistiqué	*i* / *1-i*	*1/2* / *1/2*

Figure 55 : Matrice des demandes

L'ensemble des demandes possibles i (stratégie obstinée) est supposé fini et noté $I \subset [0, 1]$. La demande r d'une stratégie responsable est noté $\{r\}$. Et enfin, la réunion de ces deux ensembles de stratégies est noté S. S unit d'une part l'ensemble de toutes les demandes possibles pour la stratégie obstinée i, et d'autre part la stratégie sophistiquée r : $S = I \cup \{r\}$.

Les stratégies obstinées telles que $i < 1/2$ sont appelées **modestes**, et les stratégies telles que $i > 1/2$ sont appelées **immodestes**. Ellingsen fait l'hypothèse que les stratégies dites **justes** ($i = 1/2$) et **avides** ($i = 1$) sont toujours comprises dans I.

La matrice suivante présente la matrice des gains p de chaque acteur lorsqu'il y a réussite de la négociation. On a alors $\pi_{ir} = i$, $\pi_{ri} = 1-i$, $\pi_{rr} = 1/2$.

	Obstiné	Sophistiqué
Obstiné	*i/(i+j)* / *j/(i+j)*	*1-j* / *j*
Sophistiqué	*i* / *1-i*	*1/2* / *1/2*

Figure 56 : Matrice des gains

En cas d'échec aucun gain ne sera attribué aux acteurs.

Ellingsen note $G = \langle S, \pi \rangle$, le jeu dans lequel chaque agent de la population exerce une stratégie $s \in S$ et dont la matrice des gains est π.

7.2.1.3 La détermination de l'équilibre évolutionnairement stable

Ellingsen cherche à déterminer l'équilibre évolutionnairement stable de la population, c'est-à-dire quelles stratégies peuvent survivre sur le long terme malgré l'introduction de populations mutantes. Soit une population de profil n, on note n_s la part de la population qui joue la stratégie s dans n. On note e_s une population dont tous les individus jouent la stratégie s. Le paiement espéré d'une stratégie s dans une population de n individus est :

$$U(e_s,n) = \sum_{s' \in S} n_{s'}.\pi_{ss'} \qquad 33)$$

$n_{s'}$: proportion de la stratégie s' dans la population

$\pi_{ss'}$: gain de la stratégie s dans sa confrontation avec la stratégie s'

Le paiement moyen d'une population n qui rencontre les membres d'une population $\tilde{n}$ est :

$$U(n,\tilde{n}) = \sum_{s \in S} \sum_{s' \in S} n_s.\tilde{n}_{s'}.\pi_{ss'} \qquad 34)$$

et le paiement moyen d'une population n est $U(n, n)$.

Par ailleurs, Ellingsen utilise la technique du **réplicateur** comme outil de sélection et de réplication des populations. Pour cela il fait l'hypothèse selon laquelle à chaque stratégie de jeu correspond un réplicateur particulier. Dans une population, chaque réplicateur est associé à un degré d'adaptabilité f_i et représente une proportion de population p_i. La dynamique du réplicateur, c'est à dire l'évolution des proportions de chaque réplicateur dans la population, est donnée par l'équation différentielle :

$$\frac{dp}{dt} = p.(f_i - \bar{f}) \text{ avec } \bar{f} \text{ le degré d'adaptabilité moyen} \qquad 35)$$

La transposition de cette équation au modèle d'Ellingsen, nous donne l'équation différentielle suivante :

$$\dot{n} = [u(e_s,n) - u(n,n)].n_s \qquad 36)$$

Pour définir l'état d'équilibre, c'est-à-dire la stratégie évolutionnairement stable, Ellingsen s'appuie sur le critère de stabilité de Lyapunov qui suppose qu'un petit choc exogène ne modifie pas l'état d'équilibre de la population[63]. Dans cette perspective, un choc est l'introduction d'une population de mutants dans la population d'équilibre. L'évolution va alors modifier la composition de la population et converger vers un nouvel équilibre. Si le nouvel équilibre est identique à celui qui précédait le choc, alors **l'équilibre est dit évolutionnairement stable**.

[63] Ce critère de stabilité ne prend en compte que la résistance de l'équilibre à un seul choc.

7.2.1.4 Les mutants

L'introduction d'un groupe mutant $\hat{n}$ en proportion ε dans une population n (appelée population incubée) forme une nouvelle population $\tilde{n}$ telle que :

$$\tilde{n} = \varepsilon.\hat{n} + (1-\varepsilon).n \qquad 37)$$

Le paiement moyen des mutants est $U(\hat{n},\tilde{n})$ et le paiement moyen de la population incubée est $U(n,\tilde{n})$. Dans cette perspective, une population est dite évolutionnairement stable si pour chaque $\hat{n} \neq n$, il existe un intervalle $\hat{E}=(0,\hat{\varepsilon})$ tel que pour chaque $\varepsilon \in \hat{E}, U(n,\tilde{n}) > U(\hat{n},\tilde{n})$. Une population est donc évolutionnairement stable si la population incubée gagne en moyenne strictement plus que toute population mutante. Ainsi, une population mutante n'aura aucune chance de pouvoir envahir la population incubée car son espérance de gain est plus faible. En développant, l'équation de stabilité, on obtient :

$$(1-\varepsilon).[u(n,n)-u(\hat{n},n)] > \varepsilon.[u(\hat{n},\hat{n})-u(n,\hat{n})] \qquad 38)$$

7.2.2 Analyse des résultats du modèle d'Ellingsen et questionnements critiques

7.2.2.1 Analyse des résultats

Ellingsen présente une analyse de son modèle en trois étapes :

- Etape 1, il analyse les conditions de stabilité de l'équilibre dans le jeu originel.
- Etape 2, il introduit une hypothèse d'incertitude probabiliste ad hoc sur la taille du gâteau, et étudie la version perturbée du modèle.
- Etape3, il relâche l'hypothèse selon laquelle toutes les stratégies sophistiquées sont implémentées à un coût nul.

Pour chaque étape, Ellingsen s'attache à montrer l'existence d'une population d'équilibre stable.

La première étape du jeu montre que seule la population d'agents dits justes est capable de participer au maintien de l'équilibre stable. En effet, recourir à la stratégie immodeste est inopérant pour deux raisons :

- la stratégie immodeste augmente le risque d'échec de la négociation et donc le risque d'un paiement nul,
- la moyenne de paiement de la population d'immodestes est nulle et cette dernière peut facilement être envahie par une population mutante.

Par conséquent, la seule population capable de donner lieu à un équilibre évolutionnairement stable est la population juste qui opte pour un partage 50%-50%. Dans ce cas, le recours à la stratégie

responsable permet d'obtenir des gains égaux à la stratégie juste, elle peut donc constituer une partie de la population stable mais à condition que sa proportion soit inférieure à *1/2* (dans le cas contraire, l'équilibre devient instable). On a donc coexistence des justes et des responsables autour d'un partage 50%-50%.

La seconde étape du jeu consiste en l'introduction d'une hypothèse ad hoc : l'incertitude probabiliste sur la taille du gâteau. Ellingsen pose la probabilité τ pour laquelle le gâteau a une taille comprise entre $1-\delta$ et *1*. Il fait également l'hypothèse que les agents sophistiqués ont connaissance de τ, alors que les obstinés n'en ont pas connaissance. Le changement principal, dans cette seconde étape du modèle, est que deux obstinés qui se rencontrent avec une demande cumulée comprise entre $1-\delta$ et *1* obtiendront un gain nul. Si l'on note $\tilde{\pi}$ la nouvelle fonction de paiement associée à la réduction δ du gâteau, le nouveau jeu perturbé est $\tilde{G}=\langle S,\tilde{\pi}\rangle$. Ellingsen montre ainsi que dans les environnements avec bruits, c'est-à-dire si le gâteau est plus petit que prévu, les comportements sophistiqués apparaissent autant que les obstinés. Et cela conduit selon Ellingsen au conflit, c'est-à-dire à une issue qui n'est pas un équilibre évolutionnairement stable. En effet, l'incertitude entraîne dans un premier temps le développement de la stratégie responsable, sur laquelle s'appuie dans un second temps l'invasion d'un groupe de mutants avides (greedy strategy). Finalement, les mutants avides sont menés à l'échec et ne constituent donc pas une population évolutionnairement stable.

Dans une troisième étape, Ellingsen introduit un coût $k>0$ de recours à la stratégie sophistiquée. Dans ce cas, la nouvelle fonction de paiement est la suivante : $\pi^k_{rs}=\pi_{rs}-k$ et $\pi^k_{ss'}=\pi_{ss'}$. Les résultats montrent que le recours à la stratégie sophistiquée étant coûteux, cette dernière ne se développe pas. De ce fait, les échecs deviennent plus fréquents du fait de la forte proportion d'immodestes dans la population. On se retrouve donc dans une configuration proche de la première étape, et in fine la seule population capable de donner lieu à un équilibre évolutionnairement stable est la population juste qui opte pour un partage 50%-50%.

Ainsi, les trois étapes du modèle montrent la convergence quasi systématique vers des stratégies d'appropriation justes, où chacun des agents tend à demander 50% du gâteau plutôt que d'adopter des stratégies sophistiquées coûteuses. Par ailleurs, le modèle montre le caractère équilibré des relations de domination dans une négociation. En effet, l'excès de domination, qui correspond aux stratégies immodestes, conduit à l'échec. On voit bien qu'une négociation débouche sur une issue favorable lorsque les agents optent pour des stratégies dites justes, qui correspondent au renoncement du recours à la domination.

7.2.2.2 Questionnements critiques

Si la démarche d'Ellingsen conduit effectivement à une lecture des stratégies de négociation en fonction des capacités de communication des joueurs, des dynamiques de gains associées à l'ultimatum, et des coûts de négociation, elle laisse de côté quatre dimensions importantes.

- La première est celle des conditions dans lesquelles sont initialisées les négociations, et sur lesquelles Ellingsen reste silencieux. Certaines conditions initiales favorisent-elles ou non les comportements de domination, ou au contraire de concession ?
- La seconde est celle de l'exercice du pouvoir dans la négociation. En effet, le pouvoir est ici un attribut des joueurs, au sens où il est contenu dans la définition même des stratégies et des profils. Une stratégie avide sera une stratégie de domination, alors qu'une stratégie modeste ou bien une stratégie responsable relèvent pour la première de la modestie et pour la seconde de la concession. Le pouvoir est par ailleurs présent dans les règles du jeu. En effet, l'ultimatum n'est autre qu'une menace coercitive qui requiert un assujettissement collectif des joueurs. Cependant se pose ici la question de l'exercice du pouvoir dans la négociation. En effet, ce dernier étant fixé par les règles, il n'est pas possible d'en faire apparaître une quelconque instrumentalisation dans le déroulement de la négociation.
- La troisième dimension est celle des temporalités associées aux dynamiques de changement de stratégies. Rien n'est dit, par exemple, sur la temporalité d'une stratégie sophistiquée lorsque celle-ci est coûteuse à maintenir dans un jeu, ou sur les temporalités qui marquent le passage d'un comportement de demande à un autre.
- Et enfin, la quatrième dimension non explorée est la façon dont le comportement de négociation affecte la taille du gâteau, et en retour comment la variation de cette taille affecte les comportements de négociation dans le temps. Plus loin, on peut se demander comment les agents vont adapter leurs stratégies en fonction de la mémoire des négociations passées. Nous tenterons, dans cette perspective, d'apporter des éléments de réponse à ces questionnements par le recours à la technique des simulations informatiques basées sur les algorithmes évolutionnaires.

7.3 Une simulation inspirée du modèle d'Ellingsen

7.3.1 L'intérêt d'une simulation

L'intérêt du recours à une simulation réside essentiellement dans l'analyse de l'évolution des comportements de négociation dans le temps, et notamment dans la persistance ou le renversement de la dynamique domination/concession. Il s'agira ainsi de simuler l'instrumentalisation du pouvoir

dans un jeu de négociation inspiré du modèle d'Ellingsen. Nous nous appuierons donc sur le jeu de négociation communicationnelle d'Ellingsen pour montrer que dès lors que l'on instrumentalise le pouvoir, il est possible d'en faire apparaître une autre dimension : le pouvoir du faible mis en évidence par T.Schelling [1960]. Or une simulation peut permettre la mise en perspective de l'exercice du pouvoir dans une négociation sous ultimatum. Il sera ainsi possible de faire apparaître des temporalités associées aux comportements de négociation, et plus particulièrement des dynamiques de phasage favorisant cycliquement ou successivement certains comportements. Il s'agira aussi de tester le rôle d'une "mémoire des négociations antérieures" dans la persistance de la stabilité ou de l'instabilité.

Ce travail s'inscrit dans le cadre des travaux menés dans le domaine de la vie artificielle, qui substitue la problématique de l'émergence des règles en environnement complexe à la problématique traditionnelle de l'équilibre. L'intérêt d'une simulation, en effet, est qu'elle s'inscrit dans une démarche cognitive articulée autour du triptyque perception/raisonnement/prise de décision [Haton 1993]. La perception renvoie aux mécanismes d'acquisition d'informations qui vont donner aux agents une représentation particulière de l'environnement à un moment donné. Le raisonnement renvoie quant à lui aux mécanismes d'apprentissage qui vont permettre l'ajustement des comportements individuels en fonction de l'état de l'environnement, c'est-à-dire en fonction des informations reçues. Le processus de décision, enfin, marque l'issue d'un raisonnement, c'est-à-dire le choix délibéré d'une action. Il s'agit donc d'un processus de recherche heuristique de la solution d'un problème spécifié, la simulation conférant aux agents une capacité d'exploration et d'évaluation de l'espace des actions possibles dans un environnement changeant. La simulation est techniquement réalisée sur la base de l'implémentation d'opérateurs d'évolution, comme le croisement et la mutation, qui rendent possible l'émergence de ce type de comportements agrégés dits "intelligents".

Dans la perspective d'apporter des éléments de réponse à nos questionnements sur le pouvoir et sur sa temporalité, nous proposons ici une adaptation du modèle d'Ellingsen autour de trois simulations progressives S1, S2 et S3 :

- **S1 - Evolution du comportement de négociation lorsque la taille du gâteau est connue**. Il s'agira de simuler le modèle d'Ellingsen dans sa version basique, afin de tester si les agents justes constituent effectivement la majorité de toute population stable, ou bien s'il émerge des stratégies particulières de domination. Nous tenterons également d'établir dans quelle mesure la composition de la population initiale influence le résultat de la simulation.
- **S2 - Evolution du comportement de négociation lorsque la taille du gâteau est inconnue**. Dans ce cas, il s'agira de tester si les sophistiqués ont effectivement l'avantage

dans une telle configuration, et si cet avantage tourne au conflit dans un second temps. Nous poserons pour cela l'hypothèse selon laquelle ni les obstinés, ni les sophistiqués n'ont connaissance de la taille réelle du gâteau[64]. Ils devront donc l'évaluer sur la base d'opérateurs de croisement et de mutation.

- **S3 - Evolution du comportement de négociation lorsque la taille du gâteau varie en fonction des comportements de négociations antérieurs**. On tentera ici de montrer comment évoluent les demandes des agents, sachant que ces dernières ont un impact certain sur la taille du gâteau. L'objectif est de faire apparaître des dynamiques de phasage des comportements de négociation. Les agents mettront-ils en place des stratégies de demandes justes visant à faire croître le gâteau pour ensuite développer des stratégies avides obstinées ? Ou bien auront-ils intérêt à osciller d'une stratégie à l'autre pour ne pas développer une situation d'échec définitif ? Comment instrumentalisent-ils le pouvoir dans la négociation ?

Les simulations S2 et S3 s'appuient sur l'hypothèse fondamentale suivant : **ni les sophistiqués ni les obstinés n'ont connaissance de la taille du gâteau**. Ils vont donc la calculer sur la base d'opérateurs d'évolution.

Cette hypothèse a une conséquence forte : les sophistiqués peuvent être conduits à l'échec, puisqu'ils doivent eux aussi évaluer la taille du gâteau.

7.3.2 Le modèle de simulation

Cette partie est consacrée à la mise en œuvre des trois simulations S1, S2 et S3. Pour cela, nous introduisons un certain nombre de modifications par rapport au modèle initial pour les simulations S2 et S3.

Dans la simulation S1, nous simulons le modèle d'Ellingsen dans sa forme de base.

Dans la simulation S2, le modèle d'Ellingsen suppose que la taille du gâteau varie à la baisse. En effet, Ellingsen pose la probabilité τ pour laquelle le gâteau a une taille comprise entre $1-\delta$ et 1. Le problème de l'incertitude est réduit dans ce cas à celui de la surestimation de la taille du gâteau, qui conduit au conflit. Cela signifie que les agents ne modifient pas leurs demandes et que leur incapacité à s'adapter conduit à l'échec généralisé, c'est-à-dire à l'absence d'équilibre évolutionnairement stable.

Dans notre modèle de simulation, la taille du gâteau est totalement inconnue et les agents doivent en faire l'apprentissage. Ils peuvent donc aussi bien se trouver dans une situation où ils surestiment

[64] Dans le modèle d'Ellingsen, seuls les obstinés ont méconnaissance de la taille réelle du gâteau. Nous complexifions donc ici le problème en posant que les sophistiqués n'ont pas connaissance de la taille du gâteau.

la taille du gâteau, que dans une situation où ils la sous-estiment. L'apprentissage porte sur la taille espérée du gâteau *teg*, dont l'évaluation conduit à une modification de leur demande *d*. Les agents sont donc dotés d'une capacité endogène à modifier leur demande *d*. Pour prendre en compte ces comportements nous avons décomposé la **demande** *d* d'un obstiné en deux composantes : **la taille du gâteau espérée** et **la portion de gâteau demandée**. Ainsi :

d = taille espérée du gâteau (teg) * portion demandée (i)

avec

T, la taille réelle du gâteau

$teg \in [0, TG]$, valeur minimale et valeur maximale de teg

$i \in I \subset [0, 1]$, I ensemble des portions demandées

donc

$d \in D \subset [0, TG]$, D ensemble fini des demandes possibles.

Dans le cas de notre simulation S2, l'essentiel n'est pas que les individus trouvent la taille exacte du gâteau mais qu'ils se rapprochent le plus possible de celle-ci afin de déboucher, en fonction de leur stratégie de demande, sur un maximum de négociations réussies donc un maximum de gain.

Dans la simulation S3, toute négociation a coût : un coût lié au temps passé à négocier, un coût d'échec de négociation, un coût d'observation de la stratégie de l'adversaire, ou bien encore un coût d'observation du gâteau. Ellingsen retient dans son modèle deux coûts principaux : le coût lié à l'incertitude sur la taille du gâteau, et le coût d'observation de la stratégie de l'adversaire. Le coût d'observation de la stratégie de l'adversaire consiste à affecter un coût *k* à la stratégie sophistiquée. Le coût lié à l'incertitude est exprimé en terme de perturbation δ associée à une probabilité τ de repartir les mains vides. Cependant, ces coûts affectent les stratégies et non les résultats de la négociation.

Lorsqu'une négociation est longue et fastidieuse, elle engendre des coûts qui viennent réduire les gains finaux : coûts liés au temps passé à négocier, coût d'observation des stratégies adversaires etc. Lorsqu'une négociation est réussie, elle entraîne des externalités positives : facilités de renégociation, confiance, coopération. Implicitement cela revient à augmenter la taille du gâteau.

Dans cette perspective, la simulation S3 consistera à prendre en considération l'effet des comportements de négociation sur la variation de la taille du gâteau *T*, et vice versa. On fait donc l'hypothèse selon laquelle la taille du gâteau varie. En effet, une stratégie avide peut avoir en retour des répercussions tellement négatives sur la variation de *T* que les agents sont conduits à réduire leurs prétentions et à adopter des comportements plus concessionnaires. Pour cela on répercute sur la taille du gâteau l'effet d'un paramètre *b*, *b* représentant le rapport du nombre de réussites sur le

nombres d'échecs à chaque période. Ainsi l'augmentation du nombre de négociations réussies a un effet positif sur la taille du gâteau (et inversement). L'introduction du paramètre *b* correspond implicitement à l'introduction d'un effet de mémoire, les agents réagissant à chaque période en fonction des échecs et des réussites de la période précédente. On introduit par ailleurs un effet de seuil.

Ce type de simulation que nous qualifierons de rebouclante, est très compliquée. En effet, par leur choix, les individus agissent directement sur les futures évaluations. Dans le cadre de notre étude, cela signifie que nous risquons de ne pas trouver d'équilibre entre nos différentes stratégies.

Tout comme dans le modèle d'Ellingsen, nos simulations sont fondées sur l'interaction entre des agents capables de reconnaître leurs stratégies mutuelles. Celles-ci sont au nombre de deux :

- la stratégie **obstinée**, lorsque la demande est indépendante de celles de l'adversaire, c'est-à-dire que l'obstiné ne reconnaît pas la stratégie de l'adversaire,
- la stratégie **sophistiquée**, lorsque l'agent adapte sa demande à la demande espérée de l'adversaire. Le sophistiqué (ou responsable) reconnaît la stratégie de l'adversaire.

A chaque période les agents sont aléatoirement regroupés par paires, chaque paire ayant pour tâche de se partager un gâteau de taille *T*. Quand deux agents se rencontrent, ils déterminent la stratégie de l'adversaire (obstinée ou sophistiquée) avec qui ils vont jouer, s'il s'agit de sophistiqués. S'il s'agit d'agents obstinés, alors ces derniers sont incapables d'identifier leur adversaire et établissent leur demande en aveugle. Ils posent donc leurs demandes respectives, en fonction de ce qu'ils ont appris ou non de l'adversaire, mais aussi en fonction des évaluations respectives concernant la taille du gâteau.

7.3.2.1 La fonction de paiement – la fonction de notation

Si le joueur *i* demande d_i et le joueur *j* demande d_j, alors le joueur *i* reçoit le paiement suivant :

$$\Pi_{ij} = \begin{cases} d_i & \text{si } d_i + d_j \le T \\ 0 & \text{sinon} \end{cases} \qquad 39)$$

Comme dans le modèle initial, si la somme de d_i et d_j excède la taille réelle du gâteau *T*, alors il y a échec de négociation, et les deux joueurs n'obtiennent aucun gain. Dans le modèle initial, Ellingsen redistribuait les surplus à chaque joueur à proportion de leurs demandes respectives. Dans notre modèle les surplus ne sont pas redistribués et sont considérés comme perdus. En effet, ils nous paraissait contradictoire de redistribuer le surplus dans des simulations de stratégies de négociation qui doivent découvrir la taille réelle du gâteau. Car en évaluant la différence entre le gain et la demande, on peut immédiatement découvrir la taille réelle du gâteau.

Les agents sophistiqués, dont la stratégie est notée *r*, sont sensés identifier la stratégie de l'adversaire et adapter leur demande à la demande espérée de l'adversaire. Donc, quand un agent

sophistiqué rencontre un obstiné qui demande d_i, il fait la demande $r = teg_r - d_i$. Néanmoins, alors que ce n'était pas le cas dans le modèle d'Ellingsen, les sophistiqués peuvent aussi être menés à l'échec s'ils surestiment la taille du gâteau. Un sophistiqué qui rencontre un obstiné obtient donc

$$\Pi_{ri} = teg_r - d_i \text{ si } teg_r \leq T \qquad 40)$$

et lorsque deux sophistiqués se rencontrent, ils obtiennent

$$\Pi_{rr} = \frac{teg_r}{2} \text{ si } \frac{teg_{r_1}}{2} + \frac{teg_{r_2}}{2} \leq T \qquad 41)$$

La stratégie $teg_r/2$ est appelée "fair strategy" ou stratégie juste, et toute stratégie pour laquelle $d_i \rightarrow teg$ est appelée "greedy strategy" ou stratégie avide.

On obtient donc la matrice des paiements suivante:

	Obstiné d_i	Sophistiqué r
Obstiné d_j	d_i d_j	teg_r-d_j d_j
Sophistiqué r	d_i teg_r-d_i	$teg_r/2$ $teg_r/2$

Figure 57 : Matrice des paiements

A partir de cette matrice des paiements, nous avons mis en place la fonction de notation d'un individu qui permet de calculer les gains espérés de cet individu en fonction de son profil.

Un échantillon de la population est choisi aléatoirement. Ensuite tous les individus de la population négocient le partage du gâteau avec chaque membre de l'échantillon. La somme des gains obtenus par l'individu représente sa fitness. La taille de cet échantillon est choisie de manière à être représentative de la population. Dans nos simulation, cette taille est égale à 10% de la taille de la population.

7.3.2.2 Les profils – le génotype

L'ensemble des stratégies est maintenant l'ensemble $S=D \cup \{r\}$, avec r la stratégie sophistiquée et D l'ensemble des demandes possibles des obstinés. Pour examiner quelles stratégies peuvent espérer survivre sur le long terme, c'est-à-dire être répliquées, on pose l'existence d'un profil n de population, avec n_s le partage effectué par la population qui joue la stratégie s. Chaque profil fera l'objet d'une notation e_s sur la base de la part relative de la population qui joue s. Ellingsen ne donne cependant pas plus d'informations sur la nature des profils, ni sur leur nombre. Nous donnerons donc un contenu à cette notion de profil en divisant la population d'agents obstinés en sept profils, qui correspondent à sept intervalles discrets compris entre [*0*, *100*] dont on choisi pour référence la valeur-centre. A l'initialisation du jeu, les agents obstinés sont répartis aléatoirement ou de façon guidé entre les sept profils obstinés.

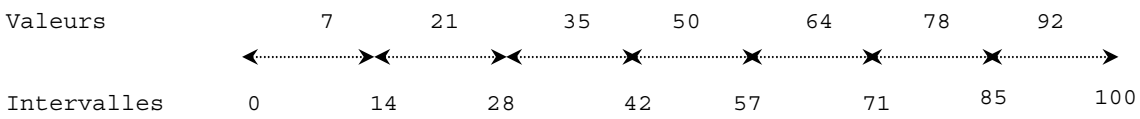


Figure 58 : Découpage de l'espace de demandes

Pour représenter les différents profils des stratégies, nous avons créé un génotype constitué de deux caractères génétiques. Le premier caractère identifie le type de stratégie de l'individu et le second caractère est la taille du gâteau espérée par l'individu.

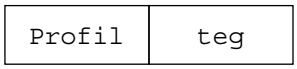


Figure 59 : Caractères génétiques d'un individu

7.3.2.3 Le déroulement des simulations

Une simulation est basé sur un algorithme génétique et se déroule selon les étapes suivantes :

1) L'initialisation correspond au tirage aléatoire ou guidé des *p* profils de populations, et donne ainsi la répartition initiale entre les différentes populations.
2) La notation correspond à l'évaluation de la fitness de tous les individus, c'est-à-dire à la notation de chaque individu en fonction de l'espérance de gains que sa stratégie est capable de générer lors de rencontres avec d'autres individus.
3) La sélection est le processus par lequel des individus sont choisis pour être répliqués, les plus favorisés étant les individus qui ont la fitness la plus élevée.
4) Le croisement correspond au mélange des individus précédemment sélectionnés pour la production de nouveaux individus mieux adaptés à l'environnement.
5) La mutation réinjecte aléatoirement un petit pourcentage de profils dans la population pour en maintenir la diversité. Il s'agit donc d'une mutation exogène.
6) retour au 2.

7.3.2.4 La sélection : le principe du réplicateur appliqué à l'algorithme génétique

La dynamique de réplication sur laquelle s'appuie Ellingsen est inspirée de la notion de sélection naturelle de Darwin, et correspond à une optimisation visant à la survie du plus fort. C'est le processus par lequel des individus dans la population sont choisis pour la reproduction. Chaque réplicateur est associé à une fitness f_i, notation d'un individu, et une proportion p_i indiquant sa proportion relative dans la population. La fitness mesure le degré d'adaptation de l'individu à son environnement. Nous posons ici l'hypothèse selon laquelle le nombre d'individus dans la population totale est fixe. Il n'y a donc pas création d'individus supplémentaires au cours des processus de négociation.

La sélection est un opérateur d'évolution centrifuge, qui, pour une population donnée, va entraîner dans le temps la convergence vers une solution résolvant le problème d'adaptabilité posé par la fonction fitness. Dans notre cas, les individus dotés d'un réplicateur, c'est-à-dire d'un profil, leur permettant une grande adaptabilité vont être reproduits en plus grand nombre dans la population à la période suivante. La taille de la population étant constante, les individus les plus forts survivent et les individus les plus faible disparaissent. On obtient ainsi une population dont la diversité et la proportion de comportement des individus ne varient plus.

7.3.2.5 Le croisement

Le croisement permet à partir de deux individus (les parents) d'obtenir deux autres individus (les enfants) dont les caractères génétiques sont un mélange des caractères des parents. L'idée fondamentale est que les enfants peuvent hériter par croisement des meilleurs gènes de leurs parents et ainsi devenir des individus mieux adaptés. Le croisement porte sur la taille espérée du gâteau, et la répétition dans le temps de cet opérateur va entraîner la convergence vers une solution potentielle. Dans nos simulations, nous appliquons un croisement restreint basé sur le profil des individus.

7.3.2.6 La mutation

La mutation, quant à elle, consiste à altérer aléatoirement un ou plusieurs caractère(s) génétique(s) d'un individu.

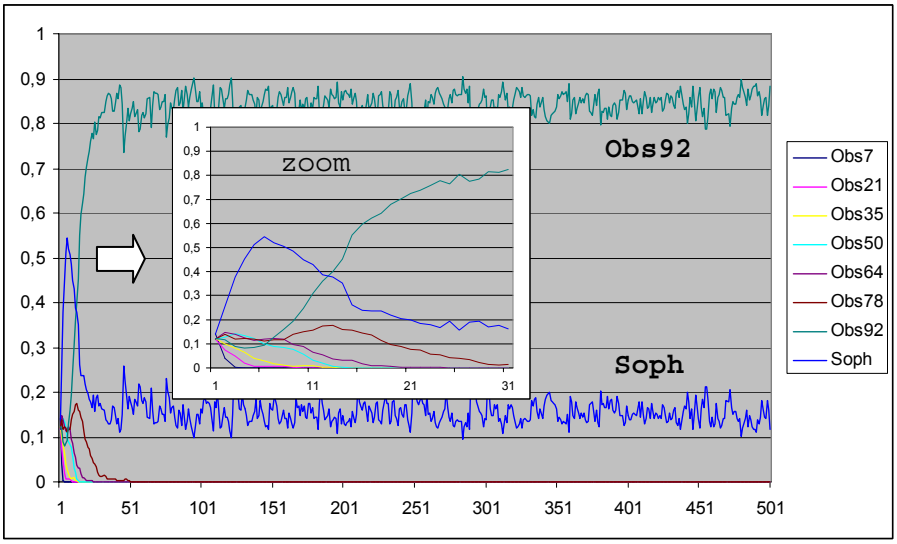
Pour la simulation S1, la mutation utilisée modifie le profil d'un individu alors que pour les simulations S2 et S3, la mutation modifie soit le profil de l'individu soit sa taille espérée du gâteau.

7.4 Résultats de la simulation

7.4.1 Simulation S1 : la taille du gâteau est connue

Il s'agit, dans cette première étape, de simuler les comportements de négociation des agents lorsque ceux-ci ont dès le départ connaissance de *T*. Les résultats de la simulation S1 infirment pour partie les résultats d'Ellingsen. En effet, ils font apparaître que le partage égalitaire peut émerger, mais uniquement si dès le départ la population était majoritairement constituée d'agents dits justes. Cette première simulation permet donc de mettre en évidence l'importance des conditions initiales dans l'évolution des comportements de négociation.

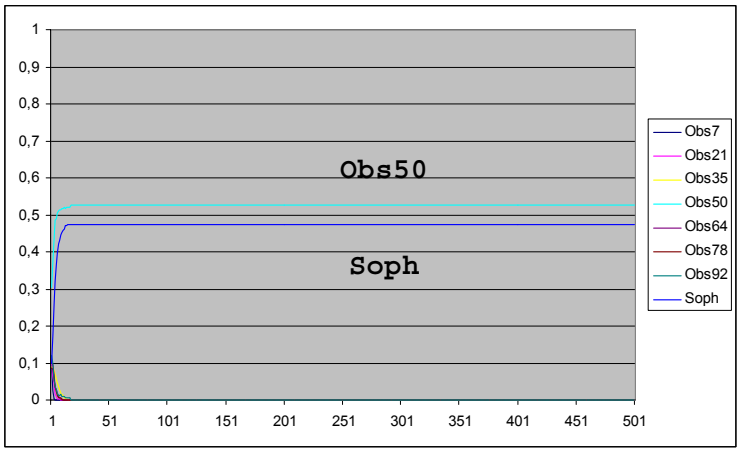
Considérons tout d'abord le cas d'une simulation où la population était au départ répartie de manière équilibrée entre les 8 profils :



Graphe 1 :Répartition équilibrée à l'initialisation

Cette première simulation montre que les comportements de négociation évoluent en deux phases distinctes. Dans un premier temps, il y a suprématie de la stratégie sophistiquée sur les autres stratégies (de la période 1 à la période 10, cf. fenêtre zoom ci-dessus). Cette suprématie des sophistiqués permet l'émergence dans un second temps (au-delà de la période 10) de la stratégie obstinée la plus dominatrice. Cette première simulation vient donc infirmer les résultats du modèle original d'Ellingsen. On montre ainsi que les forts taux d'échec conduisent dans un premier temps à des comportements concessionnaires. Ensuite ce sont ces comportements concessionnaires qui ouvrent la voie au développement de comportements ultra-dominateurs. Et l'équilibre final de négociation se stabilise autour d'une majorité d'agents obstinés dominateurs dont l'existence est maintenue du fait même de la présence d'une petite population de sophistiqués concessionnaires. Cela signifie qu'une négociation peut s'équilibrer autour d'un rapport dominant-dominé.

Testons à présent le cas où les agents dits justes sont majoritairement présents dès l'initialisation du jeu :

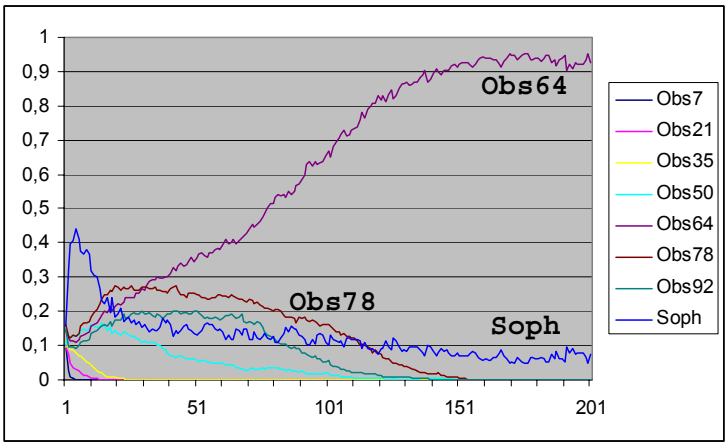


Graphe 2 : 30% d'agent "juste" à l'initialisation

On obtient donc un équilibre entre agents dits justes uniquement si la population d'obstinés dont la demande est 50% était majoritaire à l'initialisation du jeu. La stratégie 50% est adoptée par les sophistiqués et se généralise. Ce qui peut signifier du point de vue économique qu'une négociation ne débouche sur un partage égalitaire que si dès le départ les agents obstinés non dominateurs sont majoritaires dans la population. Plus loin, cela montre le rôle fondamental que jouent les conditions initiales dans le déroulement d'une négociation.

7.4.2 Simulation S2 : la taille du gâteau est inconnue

On introduit à présent de l'incertitude dans le jeu de négociation. La taille du gâteau n'est pas connue et les agents vont devoir essayer de l'estimer par croisement et par mutation. Ils peuvent tout aussi bien en surestimer qu'en sous-estimer la taille.

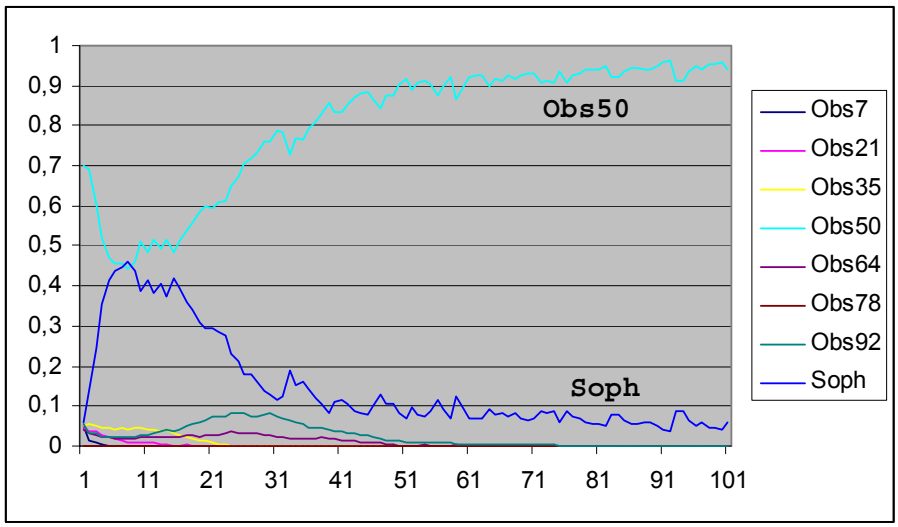


Graphe 3 :Taille du gâteau inconnue

Les résultats montrent de nouveau l'existence d'une évolution en deux phases. Une première phase courte (des périodes 1 à 15) caractérisée par l'accroissement de la population des sophistiqués, et une seconde phase caractérisée par la forte présence des obstinés. L'équilibre final s'établit ici entre les obstinés demandant 64% et les sophistiqués. Cela conduit à plusieurs remarques. En premier

lieu, l'incertitude allonge considérablement les temporalités. Alors que l'équilibre était atteint au bout d'une cinquantaine de périodes dans la simulation S1, ici il faut attendre en moyenne 200 périodes. En second lieu, les demandes espérées des obstinés sont révisées à la baisse, et la série Obs64 est à présent majoritaire dans la distribution. En effet, la possibilité d'échec étant plus forte, les agents adoptent dans un premier temps des comportements concessionnaires pour ensuite demander un peu plus de la moitié de ce qu'ils pensent être la taille du gâteau. Les stratégies sont donc plus prudentes. Les agents ont tendance à faire des concessions pour éviter le conflit, mais cela ouvre ensuite la voie à des stratégies de domination opportunistes. L'analyse du graphe correspondant à l'évolution des tailles moyennes espérées du gâteau montre l'équilibre entre les agents obstinés qui en ont légèrement surestimé la taille et les agents sophistiqués qui en ont sous-estimé la taille.

Si l'on réalise à présent une seconde simulation caractérisée par une forte présence d'agents obstinés justes à l'initialisation du jeu, on obtient de nouveau (tout comme dans S1) un équilibre entre les sophistiqués et les obstinés qui demandent 50%.

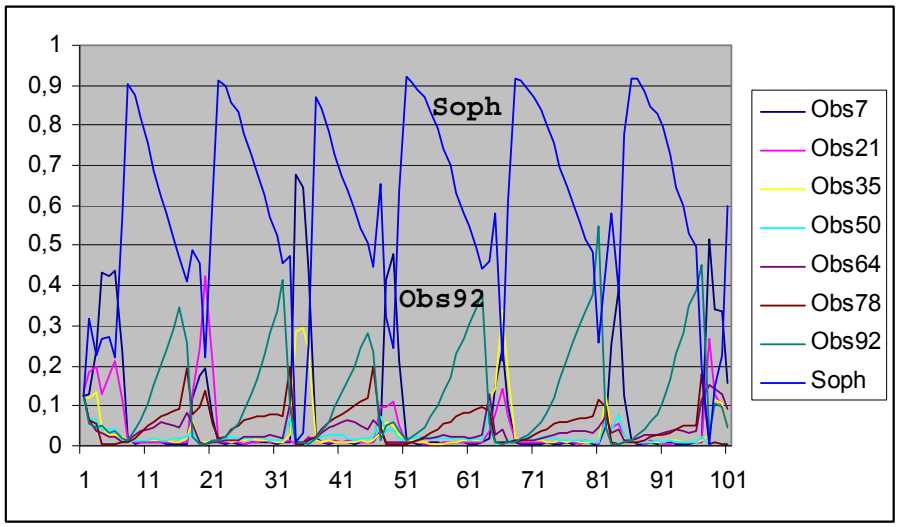


Graphe 4 :70% d'agent "juste" à l'initialisation

En résumé, contrairement au modèle d'Ellingsen qui montrait la montée en puissance des comportements avides (greedy strategies) dans le cas où la taille du gâteau n'est pas connue, nous montrons ici l'émergence de stratégies prudentes consistant en une révision à la baisse des demandes. Les concessions apparaissent massivement en début de période pour ensuite laisser place à des comportements obstinés prudents dès que la taille du gâteau est approximativement estimée.

7.4.3 Simulation S3 : le lien entre l'évolution de la taille du gâteau et le comportement de négociation

Cette troisième simulation vise à établir un lien de réciprocité entre le comportements de négociation et l'évolution de la taille du gâteau. Pour cela on répercute sur la taille du gâteau *T* l'effet d'un paramètre *b*, *b* représentant le rapport du nombre de réussites sur le nombres d'échecs à chaque période. Le résultat est le suivant :



Graphe 5 : Evolution en fonction de la taille du gâteau

Ce résultat montre que les agents passent séquentiellement d'un comportement à l'autre pour maintenir la taille du gâteau tout en satisfaisant leur intérêt individuel d'appropriation. Ainsi, on observe successivement le passage de stratégies concessionnaires à des stratégies plus dominatrices dès lors que le gâteau atteint une taille proche du seuil maximum. Ce résultat met donc en évidence la question de la flexibilité et de la dynamique de phasage des comportements dans une négociation.

Par ailleurs, on appréhende très bien le **pouvoir du faible**. En effet, la négociation est maintenue dans le temps du fait même de la présence des sophistiqués. Et pour que ces derniers soient maintenus dans le jeu, il est nécessaire que les obstinés revoient leurs prétentions à la baisse. Les obstinés ont donc tout intérêt à ce qu'il y ait une présence minimale de concessionnaires pour exercer leur domination et se maintenir dans le jeu.

7.5 Conclusions et perspectives

Les simulations S1, S2 et S3 ouvrent la voie à une réflexion centrée sur la formalisation du pouvoir appréhendé selon le rapport dominant-dominé. L'agent qui fait des concessions est beaucoup plus assuré de son éventuelle survie. Mais en retour, la domination n'est possible que si elle s'appuie sur

la présence de comportements concessionnaires, et les agents dominateurs vont mettre en place des stratégies de demande qui permettent implicitement la survie de ces derniers. C'est ce que l'on appelle ici le pouvoir du faible qui met bien en évidence l'interdépendance de joueurs en situation d'asymétrie communicationnelle.

Plus globalement, ces simulations apportent une clef de lecture des comportements de négociation dans un système agrégé de négociations bilatérales. Elles marquent une première étape dans une perspective de recherche qui se veut double. Tout d'abord, il s'agira de dépasser la limite liée à la non variabilité, dans la sélection, des stratégies obstinées/sophistiquées. Le travail à venir consistera à introduire des opérateurs de variabilité susceptibles de faire émerger des règles stratégiques nouvelles. Ensuite, ces simulations ont vocation à être appréhendées en fonction de leur portée heuristique, et par conséquent à poser un certain nombre de questions qui peuvent orienter les investigations de terrain. Il s'agira dans cette perspective d'étudier des situations de négociation sur le terrain, à la lumière des variables mises en évidence : les différentes acceptions du pouvoir, les dynamiques stratégiques de phasage, et les logiques d'appropriation.

Conclusions et perspectives

Nous avons présenté dans cette thèse les travaux de recherche sur les méthodes d'optimisation de problèmes multiobjectifs. Nous avons vu que cette problématique est abordée par la communauté scientifique suivant deux approches. La première approche tente de ramener un problème multiobjectif à un problème simple objectif au risque d'enlever toute signification au problème. La deuxième approche adopte un point de vue plus global en prenant en compte l'ensemble des critères et en utilisant la notion de dominance au sens de Pareto. Nous avons également constaté que ce domaine scientifique utilise abondamment les algorithmes évolutionnaires pour apporter une réponse à leurs problèmes. Les premières recherches ont simplement modifié l'heuristique de sélection des algorithmes génétiques. Par la suite, une heuristique de partage a été introduite pour répartir les solutions sur l'ensemble Pareto-optimal. Récemment, une population externe, appelée archive, sert à stocker les meilleurs individus. Cette archive exige la mise en place d'une stratégie de mise à jour et de réutilisation de son contenu fondée sur des heuristiques de remplacement. Les derniers travaux proposent l'utilisation de méthodes inspirées des stratégies d'évolution. Cette richesse d'idée pour augmenter l'efficience de ces méthodes a pourtant un prix : l'augmentation du nombre des paramètres de réglage et la complexification de l'écriture des algorithmes.

Ensuite nous avons présenté un domaine de recherche relativement récent qui s'intéresse à l'optimisation en environnement dynamique. La problématique issue de ce domaine est différente de celle des problèmes d'optimisation classiques. Le but n'est pas de trouver le plus rapidement ou le plus précisément possible un optimum mais d'être capable de suivre cet optimum lors de changements de l'environnement. Dans notre synthèse sur ces méthodes nous avons proposé de caractériser ces méthodes en fonction à leur capacité de détection endogène ou exogène d'un changement et, de leur approche discriminante ou non discriminante de l'espace de recherche.

Dans la deuxième partie nous avons développé une méthode multiagent, appelée la méthode des gouttes d'eau, qui permet d'optimiser une fonction multimodale en environnement dynamique. Chaque agent possède une recherche locale inspirée des stratégies d'évolution. Nous avons introduit la notion de zone d'influence qui joue le rôle d'une technique de formation de niches. Cette notion de zone d'influence, associée à cette approche multiagent, procure à notre méthode une capacité de détection endogène du changement, une capacité discriminante de l'espace de

recherche et une capacité à adapter automatiquement son nombre d'individus au problème à optimiser. De plus, nous nous sommes attachés à développer une méthode exigeant peu de paramètres : un paramètre de capacité d'exploration et un paramètre d'erreur. Les résultats encourageants obtenus avec cette méthode nous ont incité à l'étendre à des problèmes multiobjectifs.

La généralisation de notre méthode à des problèmes multiobjectifs est fondée sur l'utilisation de la notion de dominance au sens de Pareto. Nous avons modifié la définition de la zone d'influence mais le principe général de la méthode et le nombre de paramètres de réglage n'ont pas été modifiés. L'apport original de notre méthode multiobjectif est une présentation différente des solutions Pareto-optimales. Nous nous sommes également interrogé sur l'interprétation de la zone d'influence de l'agent pour un décideur.

Actuellement, notre méthode pour la résolution de problèmes multiobjectifs n'est pas capable de faire la différence entre les solutions Pareto-optimales locales et les solutions Pareto-optimales. Elle n'a pas été testée sur des problèmes multiobjectifs variés et elle n'intègre pas la notion de domination contrainte.

Nos futurs travaux de recherche essayeront d'enrichir notre méthode pour pouvoir l'appliquer à un plus large éventail de problèmes multiobjectifs. Nous souhaitons également apporter une réponse à l'interprétation de la zone d'influence.

Nous espérons aussi développer un ensemble de problèmes pour tester les méthodes en environnement dynamique. Actuellement, il n'existe aucun jeu de test validé par la communauté scientifique qui permet d'effectuer une comparaison entre les différentes méthodes.

Enfin, nous continuerons le développement de notre plate-forme de simulation en l'enrichissant de l'ensemble des techniques et des notions décrites dans cette thèse.

Bibliographie

[Allenson 1992] **Robin Allenson**, *Genetic Algorithm with Gender for Multi-Function Optimisation*, TR. EPCC-SS92-01, Edinburgh Parallel Computing Center, Edinburgh, Scotland, 1992.

[Andersen 1991] **H. C. Andersen**, *An Investigation into Genetic Algorithms, and the Relationship between Speciation and the Tracking of Optima in Dynamic Functions*, Honours thesis, Queensland University of Technology, Brisbane, Australia, 1991.

[Back 1996] **Thomas Bäck**, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New-York, 1996.

[Baricelli 1957] **Nils Aall Baricelli**, *Symbiogenetic Evolution Processes realized by Artificial Methods*, Methodos, 9, p. 143-182, 1957.

[Baricelli 1962] **Nils Aall Baricelli**, *Numerical Testing of Evolution Theories, Part II Preliminary Tests of Performance*, Symbiogenesis and terrestrial life, Acta Biotheoretica, XVI, p.99-126, 1962.

[Branke 2001] **Jürgen Branke**, *Evolutionary Approaches to Dynamic Optmization Problems - Updated survey*, GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, p. 27-30, 2001.

[Cedeno and Vemuri 1997] **W. Cedeno and V. R. Vemuri**, *On the Use of Niching for Dynamic Landscapes*, Proceedings of the 4th IEEE Int. Conf. on Evolutionary Computation (ICEC'97), IEEE Publishing, Inc., p. 361-366, 1997.

[Raphaël Cerf 1994] **Raphaël Cerf**, *Une Théorie Asymptotique des Algorithmes Génétiques*, Thèse de doctorat de l'Université Montpellier II, 1994.

[Charnes 1961] **A. Charnes and W. Cooper**, *Management Models and Industrial Applications of Linear Programming*, vol. 1, John Wiley, New-York, 1961.

[Chen and Liu 1994] **Y. L. Chen and C. C. Liu**, *Multiobjectif VAR Planning using the Goal Attainment Method*, Proceedings on Generation, Transmission and Distribution, 141, p. 227-232, 1994.

[Choquet 1953] **G. Choquet**, *Theory of capacities*, Annales de l'Institut Fourier, 5, p. 131-295, 1953.

[Cobb 1990] **Helen G. Cobb**, *An Investigation into the Use of Hypermutation as an Adaptative Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments*, TR. AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.

[Coello 1995] **Carlos A. C. Coello and al**, *Multiobjective design Optimization of Counterweight Balancing of a Robot Arm Using Genetic Algorithm*, Seventh International Conference on Tools with Artificial Intelligence, p. 20-23, 1995.

[Coello 1996] **Carlos. A. C. Coello**, *An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design*, Ph. D. thesis, Department of Computer Science, Tulane University, New Orleans, 1996.

[Coello 2001] **Carlos. A. C. Coello and Gregorio Toscano Pulido**, *Multiobjective Optimization using a Micro-genetic Algorithm*, In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), p. 274-282, San Francisco, California, 2001.

[Corne 2001] **David W. Corne, and al**, *PESA II : Region-based Selection in Evolutionary Muliobjective Optimization*, In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), p. 283-290, San Francisco, California, 2001.

[Darwin 1859] **Charles Darwin**, *The Origin of Species*, 1859.

[De Jong 1975] **Kenneth A. De Jong**, *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan, 1975.

[Deb 1999] **Kalyanmoy Deb**, *Multi-objective Genetic Algorithms : Problem Difficulties and Construction of Test Problems*, Evolutionary Computation, 7(3), p. 205-230, 1999.

[Deb 2000] **Kalyanmoy Deb**, *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multiobjective Optimization : NSGA II*, Parallel problem Solving form Nature – PPSN VI, Springer Lecture Notes in Computer Science, p. 849-858, 2000.

[Deb and al 2000] **K. Deb, S. Agrawal, S. Pratap and A. Meyarivan**, *Constrainted Test Problems for Multiobjective Evolutionary Optimization*, 2000.

[Dasgupta 1992] **Dipankar Dasgupta and D. R. Mc Gregor**. *Nonstationary Function Optimization using the Structured Genetic Algorithm*. In Proceedings of Parallel Problem Solving From Nature 2, Brussels, 28-30 September, p. 145-154, 1992.

[Duthen 1993] **Yves Duthen**, *Les projets VOXAR et IN VitrAM : Synthèse des travaux*, Habilitation à diriger des recherches, Université Paul Sabatier Toulouse III, 1993.

[Ellingsen 1997] **T. Ellingsen**, *The Evolution of Bargaining Behavior*, Quaterly Journal of economics, 112(2), p. 582-602, 1997.

[Ehrgott and Gandibleux 2000] **Matthias Ehrgott and Xavier Gandibleux**, *A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization*, OR Spektrum, 22, p. 425-460, 2000.

[Fishburn 1970] **P. C. Fishburn**, *Utility Theory for Decision Making*, John Wiley, New-York, 1970.

[Fishman 1997] **G. S. Fishman**, *Monte-Carlo : Concepts, Algorithms and Applications*, Springer-Verlarg, 1997.

[Fraser 1962] **A. S. Fraser**, *Simulation of Genetic Systems*, Journal of theorical biology, 2, p. 329-364, 1962.

[Fonseca and Fleming 1993] **Carlos M. Fonseca and Peter J. Fleming**, *Genetic Algorithm for Multiobjective Optimization : Formulation, Discussion and Generalization*, In Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, California, p. 416-423, 1993.

[Fourman 1985] **Fourman**, *Compaction of Symbolic Layout using Genetic Algorithms*. In Genetic Algorithms and their Applications : Proceedings of the First International Conference on Genetic Algorithm, p. 141-153, 1985.

[Gan and Warwick] **Justin Gan and Kevin Warwick**, *A Genetic Algorithm with Dynamic Niche Custering for Multimodal Function Optimisation*.

[Ghosh 1998] **A. Ghosh, S. Tsutsui and H. Tanaka**, *Function Optimization in Nonstationary Environment using Steady-State Genetic Algorithms with Aging of Individuals*, Proc. of the 5th IEEE Int. Conf. on Evolutionary Computation (ICEC'98), IEEE Publishing, Inc., p. 666-671, 1998.

[Glover and Laguna 1997] **F. Glover and M. Laguna**, *Tabu Search*, Kluwer Academic Publishers, 1997.

[Goldberg 1989] **David E. Goldberg**, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

[Goldberg 1989a] **David E. Goldberg**, *Sizing Populations for Serial and Parallel Genetic Algorithms*, In J. David Schaffer Ed., Proceedings of the Third International Conference on Genetic Algorithm, p. 70-79, San-Mateo, California, 1989.

[Goldberg & Richardson 1987] **Davis. E. Goldberg and J. J. Richardson**, *Genetic Algorithm with Sharing for Multimodal Function Optimisation*, Genetic Algorithms and their Applications : Proceedings of the Second ICGA, Lawrence Erlbaum Associates, Hillsdales, p. 41-49, 1987.

[Goldberg and Deb 1992] **David E. Goldberg, Kalyanmoy Deb and Jeffrey Horn**, *Massive Multimodality, Deception and Genetic Algorithms*, Ed. R. Männer, B. Manderick, Parallel Problem Solving from Nature 2, Brussels, p. 37-46, 1992.

[Grefenstette 1992] **J. J. Grefenstette**, *Genetic Algorithms for Changing Environment*, Ed. R. Maenner and B. Maanderick, Parallel Problem Solving from Nature 2, Brussels, p. 137-144, 1992.

[Hajek 1988] **B. Hajek**, *Cooling Schedules for Optimal Annealing*, Mathematics of Operations Research, vol. 13, n° 2, p. 311-329, 1988.

[Haton 1993] **J.P. Haton and M. C. Haton**, *L'Intelligence Artificielle*, Que sais-je ?, n°2444, 3e éd. corrigée, PUF, 1993.

[Harik and Lobo 1999] **G. Harik and F.Lobo**, *A Parameter-less Genetic Algorithm*, ILLIGAL TR. n° 99009, 1999.

[Holland 1975] **John Holland**, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Harbor, 1975.

[Horn and Napfliotis 1993] **J. Horn and N. Nafpliotis**, *Multiobjective Optimisation using the Niched Pareto Genetic Algorithm*, Illigal TR. n° 93005, July 1993.

[Ignizio 1981] **J. P. Ignizio**, *The Determination of a Subset of Efficient Solutions via Goal Programming*, Computing and Operations Research 3, p. 9-16, 1981.

[Ishibuchi and Murata 1996] **Hisao Ishibuchi and Tadahiko Murata**, *Multi-Objective Genetic Local Search Algorithm*. In Toshio Fukuda and Takeshi Furuhashi, editors, Proceedings of the 1996 International Conference on Evolutionary Computation, p. 119-124, Nagoya, Japan, 1996.

[Jiménez and Verdegay 1998] **F. Jiménez and J. L. Verdegay**, *Constrained Multiobjective Optimisation by Evolutionary Algorithms*, Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), p. 266-271.

[Keeney 1976] **R. L. Keeney and H. Raiffa**, *Decision with Multiple Objectives*, John Wiley, New-York, 1976.

[Koza 1992] **John R. Koza**, *Genetic Programming*, Cambridge, MA, MIT Press, 1992.

[Knowles and Corne 1999] **Joshua D. Knowles and David W. Corne**, *The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Multiobjective Optimisation*, Congress on Evolutionary Computation, p. 98-105, Washington, July 1999.

[Knowles and Corne 2000a] **Joshua D. Knowles and David W. Corne**, *Approximating the Non Dominated Front using the Pareto Archived Evolution Strategy*, Evolutionary Computation 8(2), p. 149-172, 2000.

[Knowles and Corne 2000b] **Joshua D. Knowles, David W. Corne, and Martin J. Oates**, *The Pareto-Envelope based Selection Algorithm for Multiobjective Optimization*, In Proceedings of the Sixth International Conference

on Parallel Problem Solving from Nature (PPSN VI), p. 839-848, Berlin, September 2000.

[Knowles and Corne 2000c] **Joshua D. Knowles and David W. Corne**, *M-PAES : A Memetic Algorithm for Multiobjective Optimization*, In Proceedings of the 2000 Congress on Evolutionary Computation, p. 325-332, 2000.

[Kurwase 1984] **Franck Kurwase**, *A variant of Evolution Strategies for Vector optimisation*, Ph. D Thesis, Vanderbilt University, Nashville, Tennessee, 1984.

[Kurwase 1991] **Franck Kurwase**, *A variant of Evolution Strategies for Vector Optimisation*, Parallel problem Solving from Nature First Workshop PPSN I, vol. 496 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, p. 193-197, 1991.

[Largeron et Auray] **Christine Largeron et Jean-Paul Auray**, *Proximités : approches multiformes*, chap. 2, p. 41-63.

[Levy and Montalvo 1985] **A. V. Levy and A. Montalvo**, *The Tunneling Algorithm for Global Minimization of Functions*, SIAM Journal of Scientific and Statistical Computing, vol. 6, n° 1, p. 15-29, 1985

[Lewis 1998] **J. Lewis, E. Hart, and G. Ritchie**, *A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems*. In Eiben et al. [10], p. 139-148, 1998.

[Lis and Eiben 1996] **J. Lis and A. E. Eiben**, *A Multi-Sexual Genetic Algorithm for Multiobjective Optimization*, In T. Fukuda and T. Furuhashi ed., Proceedings of the 1996 International Conference on Evolutionary Computation, Nagoya, Japan, p. 59-64, 1996.

[Luga 1997] **Hervé Luga**, *Vie artificielle et Synthèse d'images : Etude des mécanismes évolutionnistes pour la synthèse de formes et de comportements*, Thèse de doctorat de l'Université Paul Sabatier Toulouse III, 1997.

[Mahfoud 1995] **Samir W. Mahfoud**, *Niching Methods for Genetic Algorithms*, IlliGAL TR. n° 95001, University of Illinois at Urbana-Champaign, Urbana, May 1995.

[Marichal 1999a] **J. L. Marichal**, *Aggregation of Interacting Criteria by Means of the Discrete Choquet Integral*, Preprint 9910, GEMME, University of Liège, Belgium, 1999.

[Marichal 1999b] **J. L. Marichal**, *Axiomatic Foundations for a Qualitative Multicriteria Decision Making Theory*, Preprint 9920, GEMME, University of Liège, Belgium, 1999.

[Marichal 2000] **J. L. Marichal**, *L'Utilisation de l'Intégrale de Sugeno Discrète en Aide Multicritère à la Décision*, 3ième congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, Nantes, 2000.

[Messine 1997] **Frédéric Messine**, *Méthodes d'Optimisation Globale basées sur l'Analyse d'Intervalle pour la Résolution de Problèmes avec Contraintes*, Thèse de doctorat de l'Université Paul Sabatier Toulouse III, 1997.

[Moscato 1989] **Pablo Moscato**, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*, Technical Report N°826, California Institute of Technology, Pasadena, California, USA, 1989.

[Maystre 1994] **L. Y. Maystre, J. Pictet et J. Simos**, *Méthode Multicritères ELECTRE : Description, conseils pratiques et cas d'application à la gestion environnementale*, Presses polytechniques et universitaires romandes, Lausanne, 1994.

[Miller and Shaw 1995] **Brad L. Miller and Michael J. Shaw**, *Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimisation*, ILLiGAL Report N°95010, University of Illinois, December 1995.

[Minoux 1983] **Michel Minoux**, *Programmation Mathématique : Théories et Algorithmes*, Dunod, vol. 1, Paris, 1983.

[Mori 1996] **N. Mori, H. Kita and Y. Nishikawa**, *Adaptation to a Changing Environment by Means of the Thermodynamical Genetic Algorithm*, Parallel Problem Solving From Nature IV - Lecture Notes in Computer Science, vol. 1141, Springer-Verlag, p. 513-522, 1996.

[Morse 1980] **J. N. Morse**, *Reducing the Size of Nondominated Set : Pruning by Clustering*, Computers and Operation Research, 7(1-2), p. 55-66, 1980.

[Nelder and Mead 1965] **J. A. Nelder and R. Mead**, *A Simplex Method for Function Minimization*, Comput. J., vol. 7, p. 308-313, 1965.

[Pareto 1896] **Vilfredo Pareto**, *Cours d'économie politique*, vol. 1 et 2, F. Rouge, Lausanne, 1896.

[Ramsey 1993] **Connie Loggia Ramsey and John J. Grefenstette**, *Case-based Initialization of Genetic Algorithms*. In Proceedings of the Fifth International Conference on Genetic Algorithms, p. 84-91, San Mateo, California, 1993.

[Rechenberg 1973] **Ingo Rechenberg**, *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Stuttgart, 1973.

[Richardson 1989] **J. T. Richarson and al**, *Some Guidelines for Genetic Algorithms with Penalty Functions*, In J.D. Schaffer Ed., Proceedings of the Third International Conference on Genetic Algorithm, p. 191-197, 1989.

[Ritzel 1994] **B. Ritzel and al**, *Using Genetic Algorithms to Solve a Multiple Objective Groundwater Pollution Containment Problem*. Water Resources Research 30, p. 1589-1603, 1994.

[Sanza 2001] **Cédric Sanza**, *Evolution d'Entités Virtuelles Coopératives par Système de Classifieurs*, Thèse de doctorat de l'Université Paul Sabatier Toulouse III, 2001.

[Schaffer 1985] **David Schaffer**, *Multiple Objective Optimisation with Vector Evaluated Genetic Algorithm*, In genetic Algorithm and their Applications : Proceedings of the First International Conference on Genetic Algorithm, p. 93-100, 1985.

[Schoenauer et Sebag 1996] **Marc Schoenauer et Michèle Sebag**, *Contrôle d'un Algorithme Génétique*, Revue d'Intelligence Artificielle, vol. 10, n° 2-3, p. 389-428, 1996.

[Shelling 1960] **T. Shelling**, *The Strategy of Conflict*, Cambridge, Harvard University Press, 1960.

[Sugeno 1974] **M. Sugeno**, *Theory of Fuzzy Integrals and its Applications*, Ph. D., Tokyo Institute of technology, Tokyo, 1974.

[Surry 1995] **P. Surry and al**, *A Multiobjective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method*, Evolutionary Computing AISB Workshop Selected Papers, Lecture Notes in Computer Science, p. 166-180, 1995.

[Srivinas and Deb 1993] **N. Srivinas and K. Deb**, *Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms*, Technical Report, Departement of Mechanical Engineering, Institute of Technology, India, 1993,

[Tanaki 1995] **H. Tanaki and al**, *Multicriteria optimization by Genetic Algorithms : a case of scheduling in hot rolling process*, In Proceeding of the Third APORS, p. 374-381, 1995.

[Ursem 2000] **Rasmus K. Ursem**, *Multinational GA Optimization Techniques in Dynamics Environments*, Genetic and Evolutionary Computation Conference, p. 19-26, Morgan Kaufmann, 2000.

[Valenzuela and Uresti 1997] **Manuel Valenzuela and Eduardo Uresti-Charre**, *A Non-Generational Genetic Algorithm for Multiobjective Optimization*, Proceedings of the Seventh International Conference on Genetic Algorithms, p. 658-665, 1997.

[Van Veldhuizen 1999] **David A. Van Veldhuizen**, *Multiobjective, Evolutionary Algorithms : Classification, Analyses and New Innovation*, Air Force Institute of Technology, United States, 1999.

[Varak 1997a] **F. Vavak, K. Jukes and T. C. Fogarty**, *Adaptive Combustion Balancing in Multiple Burner Boiler Using a Genetic Algorithm with Variable Range of Local Search*, Proceedings of the 7th International Conference on Genetic Algorithms (ICGA'97), Michigan State University, Morgan Kaufmann Publishers, Inc., p. 719-726, 1997.

[Varak 1997b] **F. Vavak, K. Jukes and T. C. Fogarty**, *Learning the Local Search Range for Genetic Optimisation in Nonstationary Environments*, Conference on Evolutionary Computation (ICEC'97), p. 355-360, 1997.

[Vincke 1988] **Philippe Vincke**, *L'Aide Multicritère à la Décision*, Statistique et mathématiques appliquées, Ellipses, Paris, 1988.

[Xu and Louis 1996] **Z. Xu and S. J. Louis**, *Genetic Algorithms for Open Shop Scheduling and Re-Scheduling*, In Proceedings of the ISCA 11th International Conference on Computers and Their Applications., p. 99-102, Raleigh, NC, USA. International Society for Computers and Their Applications, 1996.

[Zitzler and Deb 1999] **Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele**, *Comparison of Multiobjective Evolutionary Algorithms : Empirical Results*, Tech. Report n° 70, Swiss Federal Institute of Technology (ETH), Zurich, 1999.

[Zitzler and Thiele 1998] **Eckart Zitzler and Lothar Thiele**, *An Evolutionary Algorithm for Multiobjective Optimization : The Strength Pareto Approach*, TIK-Report n° 43, 1998.

Principales références de l'auteur

[Berro 2001] **Alain Berro et Yves Duthen**, *Search for optimum in dynamic environment : a efficient agent-based method*, GECCO'2001 Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, p. 51-54, San Francisco, California, 2001.

[Berro et Leroux 2001] **Alain Berro et Isabelle Leroux**, *A simulation of bargaining behaviors : between domination and concession*, The 6th Workshop on Economics with Heterogeneous Interacting Agents, WEHIA'2001, Maastricht, 2001.

[Berro 2001] **Alain Berro et Yves Duthen**, *Recherche d'optimum d'une fonction multimodale*, 3ième journée francophones de recherche opérationnelle, FRANCORO'2001, Québec, 2001.

[Berro 2000] **Alain Berro, Stephane Sanchez et Yves Duthen**, *Optimisation par algorithme génétique du placement des succursales d'une entreprise*, 3ième congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF'2000, Nantes, 2000.

[Berro 1997] **H. Luga, R. Pelle, A. Berro et Y. Duthen**, *Extended Algebraic Surface generation for Volume Modeling : An Approach through Genetic Algorithms*, Visualization and Modelling, 1997.

Partie IV.

Annexes

Annexe A.

Rappels sur les algorithmes génétiques

A.1 De Darwin à Holland …

La découverte des principes de base de l'évolution des espèces et de ses mécanismes a été un sujet polémique entre scientifiques entre le début du XIX^{ième} siècle et le milieu du XX^{ième}.

Jusqu'à la fin du XIXième siècle la grande majorité des naturalistes admettaient que les espèces n'évoluaient pas.

Pourtant, dès 1809, Jean-Baptiste Lamarck, naturaliste français fondateur du **transformisme**, émit l'hypothèse d'une évolution des organes d'un animal en fonction de ses besoins, l'adaptation des espèces se faisant selon l'influence du milieu de vie.

En 1859, Charles Darwin, naturaliste britannique, publia son ouvrage "L'origine des espèces par voie de sélection naturelle" [Darwin 1859]. Dans son livre, il tenta de défendre le transformisme mais sans reprendre les hypothèses de lamarck. Il émit l'idée que, dans chaque espèce, la nature sélectionne les meilleurs. Chaque individu donnant naissance à beaucoup plus d'individus qu'il n'en peut survivre, cela entraîne la destruction d'un grand nombre d'entre eux et la conservation des plus aptes. C'est ce qu'il appela la sélection naturelle. Par le jeu de la sélection, les caractères avantageux se propagent peu à peu dans la population, et au bout d'un certain nombre de générations, l'espèce se trouve modifiée et mieux adaptée aux conditions d'existence. Donc les espèces se transforment. Cette théorie a été appelée le **Darwinisme** et le courant de pensée reprenant ces concepts **l'évolutionnisme**.

En 1901, De Vries, botaniste néerlandais, exposa sa théorie du **mutationnisme** selon laquelle les variations responsables de l'évolution ne se faisaient pas dans le temps mais de façon soudaine et se produisaient dans l'œuf. Il appela cela des mutations.

Aujourd'hui, on admet qu'ils avaient tous les trois en partie raison. Les bases de l'évolution étaient posées :

- L'adaptation des espèces sous la pression de leur environnement. Les individus mal adaptés disparaissent.
- La sélection des meilleurs individus. La reproduction est un combat entre individus.
- Le croisement pour assurer la pérennité de l'espèce.
- La mutation qui crée des variations génétiques néfastes ou bénéfiques pour l'individu.

D'autres chercheurs se sont intéressés aux principes biologiques des mécanismes de l'évolution.

Dès 1948, les scientifiques avaient remarqué qu'une substance colorait particulièrement bien le noyau de la cellule et faisait apparaître des filaments.

En 1866, Gregor Johann Mendel, botaniste autrichien, fut reconnu comme le fondateur de la génétique moderne. Il démontra l'existence de facteurs héréditaires par des expériences sur des petits pois.

En 1902, Walter Sutton proposa le chromosome comme site des caractères transmissibles, constituant le lieu physique de chaque caractéristique d'un être vivant.

En 1911, Johansen nomma gène l'unité élémentaire de support de l'hérédité.

En 1962, J. D. Watson, biologiste américain, F. H. Crick, biologiste britannique et M. H. F. Wilkins, biologiste britannique, reçurent le prix Nobel pour leur découverte de la structure d'A.D.N.

Ces découvertes vont être reprises et utilisées dans un autre domaine scientifique : l'informatique.

Dès 1960, de nombreux chercheurs essayèrent d'appliquer les connaissances sur l'évolution naturelle à des problèmes informatiques. [Baricelli 1957, Baricelli 1962, Fraser 1962] tentèrent de simuler l'évolution naturelle par des programmes informatiques. Mais ces tentatives furent infructueuses, probablement à cause de l'utilisation seule de la mutation.

En 1975, John Holland proposa une première solution [Holland 1975] appelée algorithme génétique[65]. Ses recherches sur les processus d'adaptation des systèmes naturels ont permis la découverte de cette nouvelle technique d'exploration. Les mécanismes de base de ces algorithmes sont très simples mais les raisons pour lesquelles cette méthode fonctionne si bien sont plus complexes et subtiles. Les fondements mathématiques des algorithmes génétiques ont été exposés par Goldberg [Goldberg 1989].

La simplicité de mise en œuvre, l'efficacité et la robustesse sont les caractéristiques les plus attrayantes de l'approche proposée par Holland. La robustesse de cette technique est due à une sélection intelligente des individus les plus proches de la solution du problème.

Nous allons maintenant voir l'algorithmique associée à cette méthode.

A.2 Fonctionnement d'un algorithme génétique

Cette section explique le fonctionnement algorithmique d'un algorithme génétique.

A.2.1 L'algorithme génétique

Un algorithme génétique commence par la création d'une première population d'individus généré aléatoirement. Ensuite l'algorithme va passer d'une génération à une autre en appliquant les mécanismes d'évaluation, sélection et modification jusqu'à l'obtention d'un critère d'arrêt.

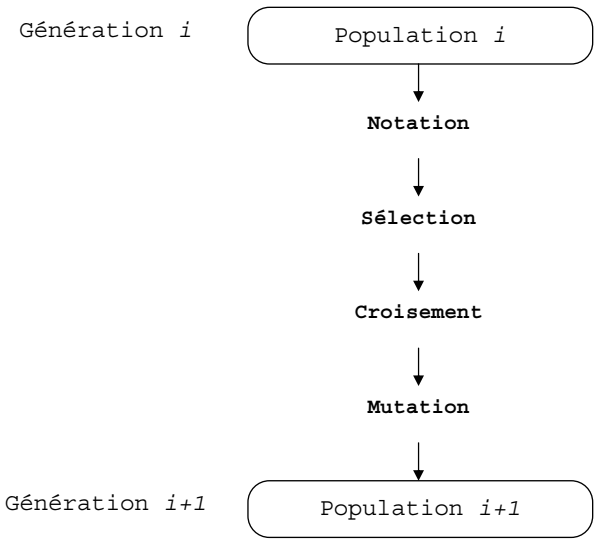


Figure 60 : Passage d'une génération *i* à *i+1*

A.2.2 L'évaluation

Dans la nature, l'adaptation d'un individu traduit sa capacité de survie dans son environnement (savoir trouver de la nourriture, échapper aux prédateurs, résister aux maladies, etc.). Cette pression exercée sans cesse sur les espèces provoque, par disparition des plus faibles, une sélection des meilleurs.

Dans la cadre d'un algorithme servant à résoudre un problème, l'adaptation d'un individu va être traduite par une mesure de sa capacité de vie qui est appelée **fitness** ou fonction de notation. Celle-ci sera définie par l'utilisateur.

A chaque début de génération, tous les individus sont notés pour permettre aux techniques de sélection de comparer les individus.

[65] En anglais : Simple Genetic Algorithm.

L'adoption d'une notation doit être faite avec soin car la convergence de l'algorithme en dépend. La notation ne doit pas être trop sévère car elle risque de privilégier les individus les plus forts et entraîner une convergence rapide qui peut s'avérer par la suite néfaste. Au début d'une simulation, les meilleurs individus sont éloignés de la solution optimale, si l'on favorise trop tôt ces individus, l'algorithme va converger vers une solution non optimale. Il ne faut pas tomber dans l'effet inverse en optant pour une notation trop lâche qui n'exercera pas une pression suffisante sur la population. Dans ce cas la convergence sera trop lente ou n'existera pas.

Plusieurs scénarii sont envisageables pour le choix d'une fonction de notation.

- Une notation fixe
- Une notation assez souple dans les premières générations dont la sévérité augmente au fur et à mesure.
- Une notation dépendante du contexte et de l'état de la population (par exemple : à l'aide de mesure statistiques). Cette notation paraît la plus efficace a priori mais elle est bien difficile à mettre en place car le paramétrage dépend essentiellement du contexte.
- Une intervention du décideur qui modifie la notation au cours du temps.

A.2.3 La Sélection

La sélection conditionne la capacité d'un individu à se reproduire. Elle décide également de la survie ou de la disparition de certains individus. Elle crée une population intermédiaire constituée de copies des individus de la population courante.

Il existe de multiples heuristiques de sélection. Nous allons voir les plus connues, la roulette pipée, le reste stochastique, le tournoi et la sélection par rang.

A.2.3.1 La roulette pipée

Le principe de la roulette pipée consiste à associer à chaque individu x une valeur proportionnelle à sa fitness. Ainsi le nombre de copies espérées pour un individu x dépend du rapport entre sa fitness $f(x)$ et la fitness moyenne $\overline{f}$ de l'ensemble des individus de la population. Donc un individu dont la note est relativement élevée par rapport à la moyenne aura plus de chance d'être sélectionné donc de transmettre ses caractères génétiques. Par contre un individu de fitness relativement faible aura un petit nombre de copies.

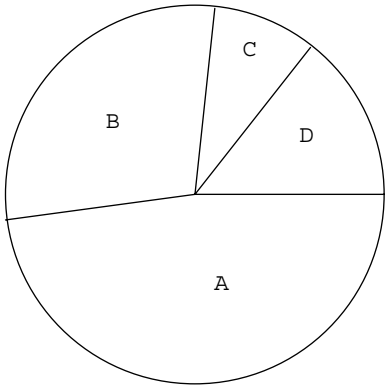


Figure 61 : Schéma de la roulette pipée

A.2.3.2 La méthode du reste stochastique

Cette méthode est utilisée à la place de la roulette pipée notamment sur des populations de petite taille[66]. Le nombre de copies espérées d'un individu x est égal à la partie entière du rapport entre sa fitness $f(x)$ et la fitness moyenne $\overline{f}$ de la population.

A.2.3.3 Le tournoi

Cette technique de sélection s'effectue en deux phases. Tout d'abord on réalise un tirage aléatoire sur l'ensemble de la population des n individus qui vont participer au tournoi. Dans cette première phase tous les individus ont la même chance d'être sélectionnés. Dans une deuxième phase on compare les notes des n individus sélectionnés pour garder le meilleur.

Cette technique de sélection est plus élitiste que les deux précédentes car la probabilité qu'un mauvais individu soit sélectionné est très faible. Mais cet élitisme peut être contrôlé en limitant le nombre de participants au tournoi. Plus ce nombre est élevé plus cette technique est élitiste.

A.2.3.4 La sélection par rang

Cette méthode classe les individus par ordre décroissant en fonction de leur note. Ainsi le meilleur individu aura une note égale à la taille de la population et le dernier aura une note égale à 1. Ensuite on applique un tirage aléatoire sur la valeur des rangs à l'aide de la roulette pipée.

La sélection par rang est souvent couplée avec une fonction de mise à l'échelle[67] qui permet de réduire ou d'augmenter l'influence des meilleurs rangs.

A.2.4 La modification

La modification d'un individu peut s'effectuer de deux manières différentes : par croisement ou par mutation.

[66] La méthode du reste stochastique est moins biaisée que la roulette pipée sur des populations de petite taille.
[67] Fonction de scaling.

Le rôle de ces opérateurs est d'explorer l'espace autour des valeurs présentes dans la population (recherche locale) mais aussi d'explorer les zones inconnues (recherche globale). Ces deux objectifs sont profondément antagonistes, car les individus n'ont pas une fonction spécifique d'exploration mais participent tous à la fois à la recherche d'une solution locale et à la recherche de solutions potentielles éloignées. Un autre antagonisme vient de l'utilisation conjuguée de la sélection et des opérateurs de modification. La sélection joue un rôle centrifuge alors que le croisement et la mutation jouent un rôle centripète[68]. Les avantages de cette capacité centrifuge/centripète sont d'assurer une certaine robustesse à la méthode.

A partir de la population des individus précédemment sélectionnés, les individus sont regroupés par paire. Chaque paire subit l'opérateur de croisement avec une probabilité t_{crois} appelée **taux de croisement**. La nouvelle population de taille n ainsi formée contient $t_{crois}*n$ nouveaux individus. Ensuite chaque individu subit l'opérateur de mutation avec une probabilité t_{mut} appelé **taux de mutation**. La population après application de la mutation constitue la génération suivante.

A.2.4.1 Le croisement

Le croisement est l'opérateur qui va permettre le brassage des caractères génétiques de la population. Cet opérateur va créer deux enfants en effectuant un mélange des chromosomes de deux parents.

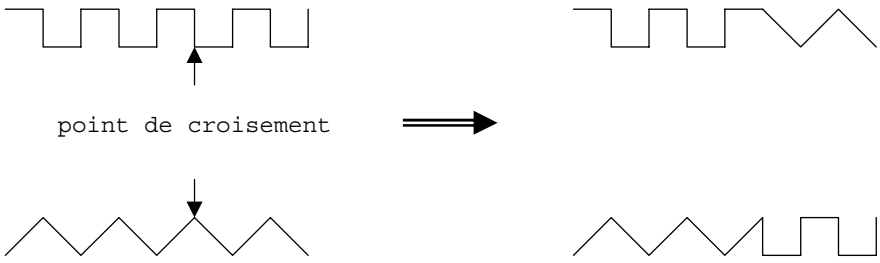


Figure 62 : Croisement avec un point de coupe

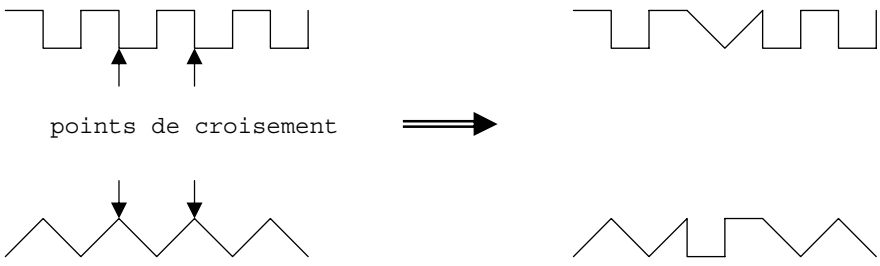


Figure 63 : Croisement avec deux points de coupe

Les figures ci-dessus illustrent deux manières possibles d'effectuer un croisement. En général la technique de croisement utilisée est très dépendante du codage de chromosomes car elle doit faire en sorte de produire des enfants génétiquement corrects.

[68] Goldberg D.E. [1989] a identifié ce dilemme sous le nom d'EVE (Exploitation versus Exploration). Cette capacité d'un algorithme génétique à converger vers une solution locale tout en recherchant une meilleure solution éloignée lui donne toute sa robustesse.

A.2.4.2 La mutation

La mutation consiste à altérer le codage d'un chromosome. Son rôle est de faire émerger de nouveaux génotypes en explorant des zones de l'espace de recherche qui pourraient ne pas être visitées par simple application de l'opérateur de croisement. Ainsi la mutation lutte contre la dérive génétique.

En pratique il existe de nombreuses manières de muter un chromosome : modification d'un ou plusieurs gène(s), changement de position d'un gène, suppression ou ajout d'un gène, …

A.2.5 Le critère d'arrêt de l'algorithme

L'arrêt de l'évolution d'un algorithme génétique est l'une des difficultés majeures car il est souvent difficile de savoir si l'on a trouvé l'optimum.

Actuellement les critères les plus utilisés sont les suivants :

- Arrêt de l'algorithme après un certain nombre de générations.
- Arrêt de l'algorithme lorsque le meilleur individu n'a pas été amélioré depuis un certain nombre de générations.
- Arrêt de l'algorithme lorsqu'il y a perte de diversité génétique.

Annexe B.

La technique de clustering

B.1 Définition

Dans certain cas de problèmes multiobjectifs, l'ensemble Pareto-optimal étant trop grand, on utilise alors une technique de clustering pour supprimer une certain nombre de solutions. Le but est d'alléger un ensemble de points tout en conservant une représentativité assez élevée [Morse 1980].Cet élagage est justifié par plusieurs raisons :

- présenter toutes les décisions aux décideurs est inutile si leur nombre est très important,
- dans le cas d'une frontière de Pareto continue, garder toutes les solutions n'est pas nécessaire,
- éviter des temps de calculs importants lorsque l'ensemble de Pareto est relativement grand.

B.2 Algorithme

L'algorithme suivant présente la technique de cluster utilisée pour la réduction d'un ensemble de Pareto-optimal.

/*----- Initialisation : chaque point est considéré comme un cluster -----*/

ensCluster = {};

Pour $i \in$ ensemble de Pareto **faire**

    ensCluster = *ensCluster* $\cup$ {*i*};

Fin Pour

/*----- Regroupement des clusters jusqu'à ce que leur nombre soit < ou = au maximum ----*/

Tant Que | *ensCluster* | > *maxParetoPoint* **faire**

    /*----- Sélection des deux clusters les plus proches -----*/

minDistance = ∞;

Pour tous les couples {*x*, *y*} de l'ensemble de Pareto **faire**

Si la distance entre *x* et *y* < *minDistance* **alors**

cluster1 = *x*;

cluster2 = *y*;

minDistance = distance entre *x* et *y*;

Fin Si

Fin Pour

/*----- Regroupement des deux clusters -----*/

nouveauCluster = *cluster1* ∪ *cluster2*;

ensCluster = *ensCluster* \ {*cluster1*, *cluster2*};

ensCluster = *ensCluster* ∪ {*nouveauCluster*};

Fin Tant Que

/*----- Calcul du centre de chaque cluster -----*/

ensPareto = {};

Pour *c* ∈ *ensCluster* **faire**

i = CalculBarycentreCluster(*c*);

ensPareto = *ensPareto* ∪ {*i*};

Fin Pour