Module Master M2

Systèmes complexes
Chapitre IV : Les Métaheuristiques

Projet n°02 d'exposé des étudiants de Master M2 Informatique

Présenté par : Prof. Kholladi Mohamed-Khireddine
Département d'Informatique
Facultés des Sciences Exactes
Université Echahid Hamma Lakhdar d'El Oued
Tél. 0770314924
Email. kholladi@univ-eloued.dz et kholladi@yahoo.fr
Site Web. www.univ-eloued.dz
http://kholladi.doomby.com/ et http://kholladi.e-monsite.com/

IV – Les Métaheuristiques

IV.0 - Sommaire

IV.1 – Introduction

Les métaheuristiques constituent une famille d'algorithmes d'optimisation également appelés algorithmes d'approximation. Elles visent à résoudre des problèmes d'optimisation difficile issus de la recherche opérationnelle pour lesquels on ne connaît pas de méthode classique plus efficace. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum par échantillonnage d'une fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème pour en trouver une approximation de la meilleure solution. L'échantillonnage est la sélection d'une partie dans un tout. Il s'agit d'une notion importante en métrologie, lorsqu'on

ne peut pas saisir un événement dans son ensemble. Il faut effectuer des mesures en nombre fini pour représenter l'événement.

Le terme prend un sens précis dans certains domaines :

1. En analyse chimique, lorsqu'on analyse un produit, une des questions qui se pose est celle de l'homogénéité. Les méthodes d'analyse ne permettent d'avoir la composition que d'une quantité finie de produit (en général, de quelques microgrammes à quelques centaines de grammes), lorsque la quantité est importante (fût, cuve, soute, tas de gravier, etc.), la composition peut varier d'un endroit à l'autre (hétérogénéité, stratification, décantation, etc.). Lorsqu'il n'est pas possible d'homogénéiser, il faut donc effectuer des prélèvements en plusieurs endroits, selon un protocole précis.
2. En archéologie, l'échantillonnage consiste à prélever une partie seulement du matériel archéologique d'un site pendant une prospection.
3. En géochimie, lorsqu'on prélève un fragment de roche, on ne sait pas quelle portion du terrain elle représente, tant en surface qu'en profondeur. On effectue alors l'échantillonnage à plusieurs endroits pour prendre en compte l'hétérogénéité du terrain.
4. La technique de sondage, utilisée en particulier en sciences humaines mais pas seulement, consiste à établir des mesures statistiques sur un échantillon seulement de la population étudiée et à en étendre l'interprétation à la population elle-même. Les conditions de validité de cette extension sont établies par la discipline mathématique des statistiques et des probabilités.
5. En traitement du signal, l'échantillonnage consiste à transformer un signal continu en signal discret, c'est le découpage temporel de l'étape de numérisation.
6. En musique, l'échantillonnage ou sampling, est la numérisation de documents sonores ou samples (échantillons), Et désigne aussi l'utilisation de samples dans la création de nouvelles compositions artistiques, ce qui peut permettre de renouveler le matériau sonore.
7. En architecture navale, l'échantillonnage désigne l'épaisseur de matériau utilisée pour les différentes pièces de la coque (bordé, renforts).Il détermine directement la résistance du navire.

Il existe un grand nombre de métaheuristiques différentes, allant de la simple recherche locale à des algorithmes complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents.

Les métaheuristiques (M) sont souvent des algorithmes utilisant un échantillonnage probabiliste. Elles tentent de trouver l'optimum global (G) d'un problème d'optimisation difficile (avec des discontinuités (D), par exemple), sans être piégé par les optimums locaux (L) comme sur la figure IV.1.

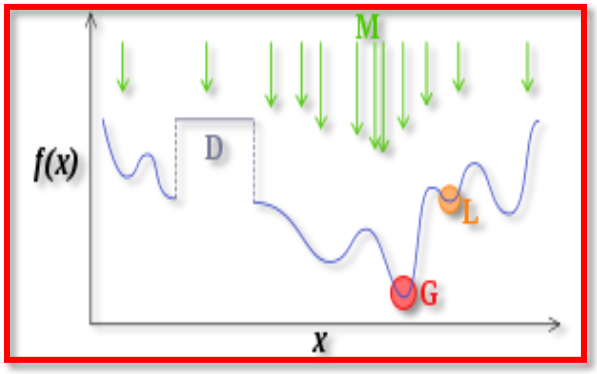

Figure IV.1 – Recherche d'optimum global

IV.2 - Généralités

IV.2.1 - Terminologies

On parle de méta, du grec "au-delà" (comprendre ici "à un plus haut niveau"), heuristique, du grec ευρισκειν / heuriskein, qui signifie "trouver". En effet, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. Une terminologie légèrement différente considère que les métaheuristiques sont une forme d'algorithmes d'optimisation stochastique, hybridés avec une recherche locale. Le terme méta est donc pris au sens où les algorithmes peuvent regrouper plusieurs heuristiques. On rencontre cette définition essentiellement dans la littérature concernant les algorithmes évolutionnaires, où elle est utilisée pour désigner une spécialisation.

Dans le cadre de la première terminologie, un algorithme évolutionnaire hybridé avec une recherche locale sera plutôt désigné sous le terme d'algorithme mimétique. Les métaheuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris

- En physique (cas du recuit simulé);
- En biologie de l'évolution (cas des algorithmes génétiques) ou encore;
- En éthologie (cas des algorithmes de colonies de fourmis ou de l'optimisation par essaims particulaires).

IV.2.2 - Nomenclature

Le but d'une métaheuristique est de résoudre un problème d'optimisation donné. Elle cherche un objet mathématique (une permutation, un vecteur, etc.) minimisant (ou maximisant) une fonction objectif, qui décrit la qualité d'une solution au problème. L'ensemble des solutions possibles forme l'espace de recherche. L'espace de recherche est au minimum borné, mais peut être également limité par un ensemble de contraintes.

Exemple de front de Pareto dans un problème nécessitant la minimisation de deux objectifs (f1 et f2). Les points A et B sont non dominés alors que le point C n'est optimum pour aucun des objectifs comme sur la figure IV.2.

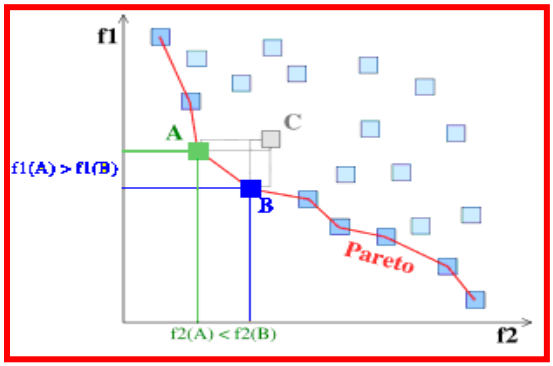

Figure IV.2 – Front de Pareto

Les métaheuristiques manipulent une ou plusieurs solutions, à la recherche de l'optimum, la meilleure solution au problème. Les itérations successives doivent permettre de passer d'une solution de mauvaise qualité à la solution optimale. L'algorithme s'arrête après avoir atteint un critère d'arrêt, consistant généralement en l'atteinte du temps d'exécution imparti ou en une précision demandée. Une solution ou un ensemble de solutions est parfois appelé un état, que la métaheuristique fait évoluer via des transitions ou des mouvements. Si une nouvelle solution est construite à partir d'une solution existante, elle est sa voisine. Le choix du voisinage et de la structure de données le représentant peut être crucial. Lorsqu'une solution est associée à une seule valeur, on parle de problème mono objectif, lorsqu'elle est associée à plusieurs valeurs, de problème multi-objectifs (ou multicritères). Dans ce dernier cas, on recherche un ensemble de solutions non dominées (le front de Pareto), solutions parmi lesquelles on ne peut décider si une solution est meilleure qu'une autre, aucune n'étant systématiquement inférieure aux autres sur tous les objectifs. Dans certains cas, le but recherché est explicitement de trouver un ensemble d'optimums satisfaisants. L'algorithme

doit alors trouver l'ensemble des solutions de bonne qualité, sans nécessairement se limiter au seul optimum. On parle de méthodes multimodales.

IV.2.3 - Concepts généraux

Comportement d'une métaheuristique : Le graphique représente les distributions des valeurs des optimums trouvés (sur un grand nombre d'exécutions) : l'algorithme passe d'une population de solution très dispersée (A) à une population plus centrée sur l'optimum trouvé (B) comme sur la figure IV.3.

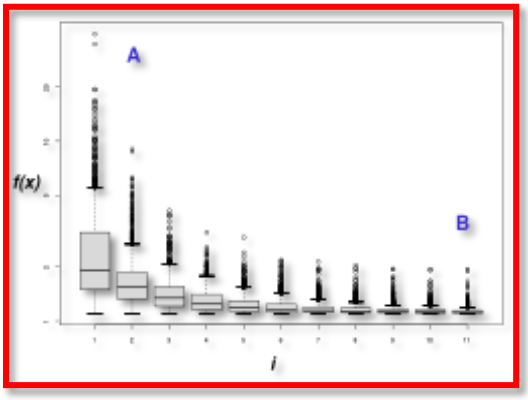

Figure IV.3 – Comportement d'une métaheuristique

Les métaheuristiques ne nécessitent pas de connaissances particulières sur le problème optimisé pour fonctionner, Le fait de pouvoir associer une (ou plusieurs) valeurs à une solution est la seule information nécessaire. En pratique, elles ne devraient être utilisées que sur des problèmes ne pouvant être optimisés par des méthodes mathématiques utilisées en lieu et place d'heuristiques spécialisées, elles montrent généralement de moins bonnes performances. Les métaheuristiques sont souvent employées en optimisation combinatoire, mais on en rencontre également pour des problèmes continus ou mixtes (problèmes à variables discrètes et continues). Certaines métaheuristiques sont théoriquement convergentes sous certaines conditions. Il est alors garanti que l'optimum global sera trouvé en un temps fini, la probabilité de ce faire augmentant asymptotiquement avec le temps. Cette garantie revient à considérer que l'algorithme se comporte au pire comme une recherche aléatoire pure (la probabilité de tenter toutes les solutions tendant vers un. Cependant, les conditions nécessaires sont rarement vérifiées dans le cadre d'applications réelles. En pratique, la principale condition de convergence est de considérer que l'algorithme est ergodique (qu'il peut atteindre n'importe quelle solution à chaque mouvement), Mais on se satisfait souvent

d'une quasi-ergodicité (si la métaheuristique peut atteindre n'importe quelle solution en un nombre fini de mouvements).

IV.2.4 - Organisation générale

Les trois phases d'une métaheuristique itérative. Les points rouges dans la figure IV.4 représentent l'échantillonnage de la fonction objectif (ici à une dimension).

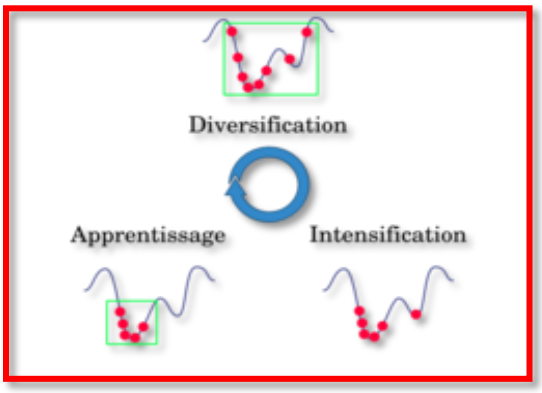

Figure IV.4 – Les trois phases d'une métaheuristique itérative

D'une manière générale, les métaheuristiques s'articulent autour de plusieurs notions de voisinage, de diversification/exploration, d'intensification/exploitation, de mémoire et d'apprentissage.

a – Voisinage

La notion de voisinage est sans doute le principe général le plus utilisé pour la conception d'heuristiques. Pour les problèmes combinatoires, le voisinage a un impact important sur le comportement des métaheuristiques, alors que pour des problèmes continus, la notion même de voisinage est plus difficile à cerner. Bien qu'il n'existe que très peu de résultats théoriques sur l'adéquation entre un voisinage et un problème discret donné, il peut être possible d'en calculer des indicateurs empiriques, comme la rugosité. Les techniques les plus classiques concernant la définition d'un voisinage tournent autour des notions de permutations, de chaînes d'éjections et d'optimisations partielles.

b - Intensification, diversification, apprentissage

Les notions d'intensification et de diversification sont liées à l'utilisation de la fonction objective et aux processus aléatoires. Combinés avec la notion de mémoire, elles permettent de positionner les différents aspects des métaheuristiques entre eux comme sur la figure IV.5

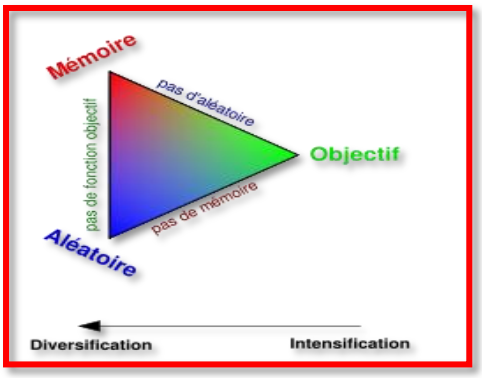

Figure IV.5 – Notions de d'intensification et de diversification

La diversification (ou exploration, synonyme utilisé presque indifféremment dans la littérature des algorithmes évolutionnaires) désigne les processus visant à récolter de l'information sur le problème optimisé. L'intensification (ou exploitation) vise à utiliser l'information déjà récoltée pour définir et parcourir les zones intéressantes de l'espace de recherche. La mémoire est le support de l'apprentissage, qui permet à l'algorithme de ne tenir compte que des zones où l'optimum global est susceptible de se trouver, évitant ainsi les optima locaux. Les métaheuristiques progressent de façon itérative, en alternant des phases d'intensification, de diversification et d'apprentissage ou en mêlant ces notions de façon plus étroites. L'état de départ est souvent choisi aléatoirement, l'algorithme se déroulant ensuite jusqu'à ce qu'un critère d'arrêt soit atteint. Les notions d'intensification et de diversifications sont prépondérantes dans la conception des métaheuristiques, qui doivent atteindre un équilibre délicat entre ces deux dynamiques de recherches. Les deux notions ne sont donc pas contradictoires, mais complémentaires et il existe de nombreuses stratégies mêlant à la fois l'un et l'autre des aspects.

IV.3 – Classification

IV3.1 - Parcours et Population

Les métaheuristiques les plus classiques sont celles fondées sur la notion de parcours. Dans cette optique, l'algorithme fait évoluer une seule solution sur l'espace de recherche à chaque itération. La notion de voisinage est alors primordiale. Les plus connues dans cette classe sont le recuit simulé, la recherche avec tabous, la recherche à voisinage variable, la méthode GRASP ou encore les méthodes de bruitage. Dans cette classification, l'autre approche utilise la notion de population. La métaheuristique manipule un ensemble de solutions en parallèle, à chaque itération (voir la figure IV.6).

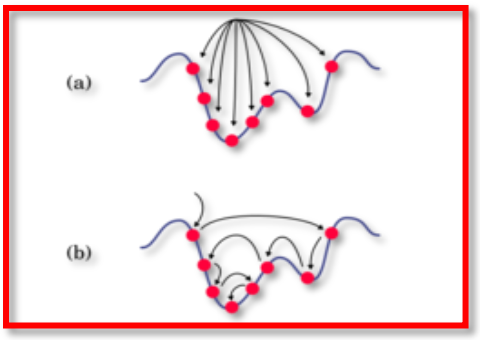

Figure IV.6 - Principe général des métaheuristiques (a) à population, et (b) à parcours.

On peut citer :

- Les algorithmes génétiques;
- L'optimisation par essaims particulaires;
- Et les algorithmes de colonies de fourmis.

La frontière est parfois floue entre ces deux classes. On peut ainsi considérer qu'un recuit simulé où la température baisse par paliers, a une structure à population. En effet, dans ce cas on manipule un ensemble de points à chaque palier, il s'agit simplement d'une méthode d'échantillonnage particulière.

IV.3.2 - Emploi de la mémoire

Les métaheuristiques utilisent l'historique de leur recherche pour guider l'optimisation aux itérations suivantes. Dans le cas le plus simple, elles se limitent à considérer l'état de la recherche à une itération donnée pour déterminer la prochaine itération : il s'agit alors d'un processus de décision markovien, et on parlera de méthode sans mémoire. C'est le cas de la plupart des méthodes de recherche locale. Beaucoup de métaheuristiques utilisent une mémoire plus évoluée, que ce soit sur le court terme (solutions visitées récemment, par exemple) ou sur le long terme (mémorisation d'un ensemble de paramètres synthétiques décrivant la recherche).

IV.3.3 - Fonction objective statique ou dynamique

La plupart des métaheuristiques utilisent la fonction objective en l'état, et font évoluer leur comportement de recherche de l'optimum. Cependant, certains algorithmes, comme la recherche locale guidée, modifient la représentation du problème, en incorporant l'information collectée durant la recherche, directement au sein de la fonction objective. Il est donc possible de classer les métaheuristiques selon qu'elles utilisent une fonction objective statique (qui

demeure inchangée tout au long de l'optimisation) ou dynamique (quand la fonction objective est modifiée au cours de la recherche).

IV.3.4 - Nombre de structures de voisinage

La plupart des métaheuristiques utilisées dans le cadre des problèmes d'optimisation combinatoire utilisent une seule structure de voisinage. Cependant, des méthodes comme la recherche à voisinage variable permettent de changer de structure en cours de recherche.

IV.3.5 - Implicite, explicite et directe

En considérant les métaheuristiques comme des méthodes itératives utilisant un échantillonnage de la fonction objectif comme base d'apprentissage (définition plus particulièrement adaptée aux métaheuristiques à populations) apparaît le problème du choix de l'échantillonnage. Dans la très grande majorité des cas, cet échantillonnage se fait sur une base aléatoire, et peut donc être décrit via une distribution de probabilités. Il existe alors trois classes de métaheuristiques, selon l'approche utilisée pour manipuler cette distribution voir la figure IV.7.

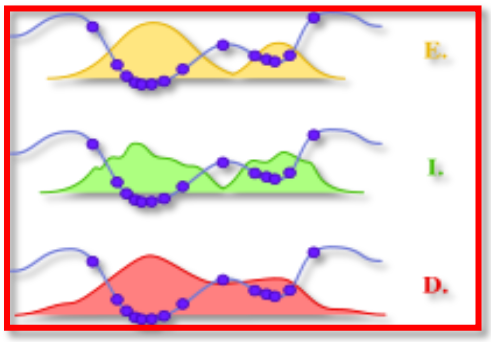

Figure IV.7 - Les métaheuristiques peuvent être qualifiées d'explicites (E, ici sur une somme de deux gaussiennes), d'implicites (I) ou de directes (D), selon la façon dont elles gèrent la transition entre deux itérations

La première classe est celle des méthodes implicites, où la distribution de probabilité n'est pas connue a priori. C'est le cas par exemple des algorithmes génétiques, où le choix de l'échantillonnage entre deux itérations ne suit pas une loi donnée, mais est fonction de règles locales. Par opposition, on peut donc classer les méthodes explicites, qui utilisent une distribution de probabilité choisie à chaque itération. C'est le cas des algorithmes à estimation de distribution. Dans cette classification, le recuit simulé occupe une place particulière, puisqu'on peut considérer qu'il échantillonne la fonction objective en utilisant directement

celle-ci comme distribution de probabilité (les meilleures solutions ayant une probabilité plus grande d'être tirées). Il n'est donc ni explicite ni implicite, mais plutôt direct.

IV.3.6 - Evolutionnaire ou non

On trouve parfois une classification présentant les algorithmes d'optimisations stochastiques comme étant évolutionnaires (ou évolutionnistes) ou non. L'algorithme sera considéré comme faisant partie de la classe des algorithmes évolutionnaires s'il manipule une population via des opérateurs, selon un algorithme général donné. Cette façon de présenter les métaheuristiques dispose d'une nomenclature adaptée : on parlera d'opérateurs pour toute action modifiant l'état d'une ou plusieurs solutions. Un opérateur construisant une nouvelle solution sera dénommé générateur, alors qu'un opérateur modifiant une solution existante sera appelé mutateur. Dans cette optique, la structure générale des algorithmes évolutionnaires enchaîne des étapes de sélection, de reproduction (ou croisement), de mutation et enfin de remplacement. Chaque étape utilise des opérateurs plus ou moins spécifiques.

IV.3.7 - Taxinomie de l'hybridation (voir la figure IV.8)

On parle d'hybridation quand la métaheuristique considérée est composée de plusieurs méthodes se répartissant les tâches de recherche. La taxinomie des métaheuristiques hybrides se sépare en deux parties : une classification hiérarchique et une classification plate. La classification est applicable aux méthodes déterministes aussi bien qu'aux métaheuristiques.

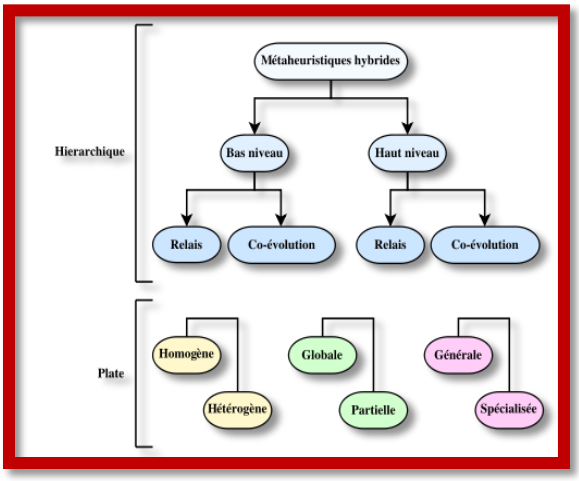

Figure IV.8 - Taxinomie des métaheuristiques hybrides

La classification hiérarchique se fonde sur le niveau (bas ou haut) de l'hybridation et sur son application (en relais ou concurrente). Dans une hybridation de bas niveau, une fonction donnée d'une métaheuristique (par exemple, la mutation dans un algorithme évolutionnaire)

est remplacée par une autre métaheuristique (par exemple une recherche avec tabou). Dans le cas du haut niveau, le fonctionnement interne normal des métaheuristiques n'est pas modifié. Dans une hybridation en relais, les métaheuristiques sont lancées les unes après les autres, chacune prenant en entrée la sortie produite par la précédente.

Dans la concurrence (ou coévolution), chaque algorithme utilise une série d'agents coopérants ensembles. Cette première classification dégage quatre classes générales :

- Bas niveau & relais (abrégé LRH en anglais);
- Bas niveau & coévolution (abrégé LCH);
- Haut niveau & relais (HRH);
- Haut niveau & coévolution (HCH).

La seconde partie dégage plusieurs critères qui peuvent caractériser les hybridations :

- Si l'hybridation se fait entre plusieurs instances d'une même métaheuristique, elle est homogène, sinon, elle est hétérogène.
- Si les méthodes recherchent dans tout l'espace de recherche, on parlera d'hybridation globale, si elles se limitent à des sous-parties de l'espace, d'hybridation partielle ;
- Si les algorithmes mis en jeu travaillent tous à résoudre le même problème, on parlera d'approche générale, s'ils sont lancés sur des problèmes différents, d'hybridation spécialisée.

Ces différentes catégories peuvent être combinées, la classification hiérarchique étant la plus générale.

IV.4 – Applications

Les métaheuristiques sont souvent employées pour leur facilité de programmation et de manipulation. Elles sont en effet facilement adaptables à tout type de problème d'optimisation. Toutefois, elles sont le plus judicieusement employées sur des problèmes d'optimisation difficile, où des méthodes d'optimisation plus classiques (méthodes déterministes, notamment) montrent leurs limites. De façon générale, on peut considérer que des problèmes présentant les caractéristiques suivantes sont assez propices à l'utilisation de métaheuristiques :

- NP-complétude,
- nombreux optima locaux,

- discontinuités,
- contraintes fortes,
- non dérivabilité,
- temps de calcul de la fonction objectif prohibitif,
- solution approchée souhaitée,
- etc.

IV.4.1 - Tests

Pour tester une métaheuristique, une première étape consiste à utiliser des fonctions mathématiques spécialement conçues. Les algorithmes sont évalués sur la base d'un ensemble de fonctions, plus ou moins difficiles, puis comparés entre eux. Les métaheuristiques étant généralement stochastiques, les tests doivent en principe être répétés un grand nombre de fois, puis exploités via des méthodes statistiques. Cependant, cette pratique reste relativement peu répandue dans la littérature spécialisée (voir la figure IV.9).

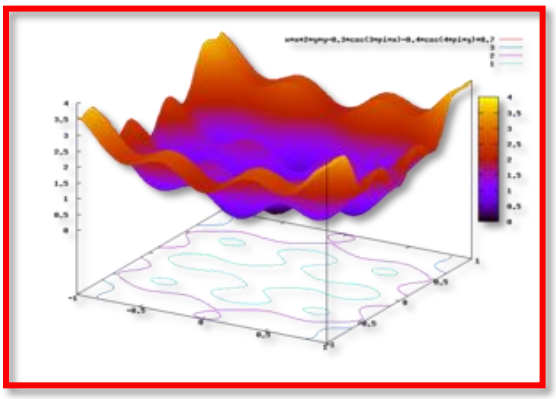

Figure IV.9 - Exemple de problème test à minimiser (présenté ici pour deux variables)

IV.4.2 - Problèmes réels

Quelques exemples de problèmes concrets, optimisés via des métaheuristiques :

1. Problèmes de tournée de véhicules;
2. Problèmes d'ordonnancements de tâches;
3. Optimisation de réseaux mobiles **UMTS**;
4. Gestion du trafic aérien;
5. Optimisation des plans de chargement des cœurs de réacteurs nucléaires;
6. Etc.

IV.4.3 - Avantages et inconvénients

Les métaheuristiques étant très généralistes, elles peuvent être adaptées à tout type de problème d'optimisation pouvant se réduire à une boîte noire. Elles sont souvent moins puissantes que des méthodes exactes sur certains types de problèmes. Elles ne garantissent pas non plus la découverte de l'optimum global en un temps fini. Cependant, un grand nombre de problèmes réels ne peut pas être optimisé efficacement par des approches purement mathématiques, les métaheuristiques peuvent alors être utilisées avec profit. La notion d'efficacité se rapporte généralement à deux objectifs contradictoires : la vitesse et la précision. La vitesse est souvent mesurée en nombre d'évaluations de la fonction objectif, qui est la plupart du temps la partie la plus gourmande en temps de calcul. La précision se rapporte à la distance entre l'optimum trouvé par la métaheuristique et l'optimum réel, soit du point de vue de la solution, soit de celui de la valeur. Bien souvent, un algorithme rapide est peu précis, et inversement. Généralement, un choix doit être fait quant au critère d'arrêt adéquat. Un nombre d'évaluation ou un temps imparti est souvent utilisé, mais on peut également choisir d'atteindre une valeur donnée de la fonction objective (le but étant alors de trouver une solution associée). Il est également possible de choisir des critères dépendants du comportement de l'algorithme, comme une dispersion minimale de la population de points ou un paramètre interne approprié. En tout état de cause, le choix du critère d'arrêt influencera la qualité de l'optimisation.

L'utilisation de métaheuristiques peut paraître relativement simple, en première approche, mais il est souvent nécessaire d'adapter l'algorithme au problème optimisé. Tout d'abord, principalement dans le cadre de l‘optimisation combinatoire, le choix de la représentation des solutions manipulées peut être crucial. Ensuite, la plupart des métaheuristiques disposent de paramètres dont le réglage n'est pas nécessairement trivial. Enfin, obtenir de bonnes performances passe généralement par une étape d'adaptation des diverses étapes de l'algorithme (initialisation, notamment). En pratique, seul le savoir-faire et l'expérience de l'utilisateur permettent de gérer ces problèmes.

Le théorème du "no free lunch" explique qu'aucune instance de métaheuristique ne peut prétendre être la meilleure sur tous les problèmes. Une métaheuristique (M) n'est performante que pour une classe de problème (P) donnée (voir la figure IV.10).

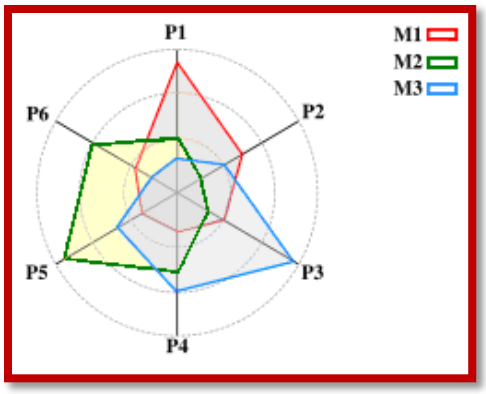

Figure IV.10 - Le théorème du "no free lunch"

Il est admis que, d'un point de vue très général, aucune métaheuristique n'est réellement meilleure qu'une autre. En effet, une métaheuristique ne peut prétendre être plus efficace sur tous les problèmes, bien que certaines instances (c'est-à-dire l'algorithme lui même, mais aussi un choix de paramètres et une implémentation donnée) puissent être plus adaptées que d'autres sur certaines classes de problèmes. Cette constatation est décrite par le théorème du no free lunch ("pas de dîner gratuit").

En dernière analyse, il est parfois possible que le choix de la représentation des solutions, ou plus généralement des méthodes associées à la métaheuristique, ait plus d'influence sur les performances que le type d'algorithme lui-même. En pratique, cependant, les métaheuristiques se montrent plus puissantes que les méthodes de parcours exhaustif ou de recherche purement aléatoire.

IV.5 – Variantes

IV.5.1 - Liste de métaheuristiques

Ici, on dresse une liste de métaheuristiques suivante :

1. Les métaheuristiques les plus connues sont :
2. Les algorithmes évolutionnaires (comprenant les algorithmes génétiques),
3. le recuit simulé,
4. la recherche avec tabous,
5. les algorithmes de colonies de fourmis,
6. Les algorithmes d'optimisation par essaims particulaires,
7. les algorithmes à estimation de distribution.

Il existe un très grand nombre d'autres métaheuristiques, plus ou moins connues :

1. L'algorithme du kangourou;
2. La méthode de Fletcher et Powell;
3. La méthode GRASP;
4. La méthode du bruitage;
5. Les systèmes immunitaires artificiels;
6. Les algorithmes à évolution différentielle;
7. La tunnelisation stochastique;
8. L'escalade de collines à recommencements aléatoires;
9. Etc.

La recherche dans le domaine étant très active, il est impossible de produire une liste exhaustive des différentes métaheuristiques d'optimisation. La littérature spécialisée montre un grand nombre de variantes et d'hybridations entre méthodes, particulièrement dans le cas des algorithmes évolutionnaires**.**

IV.5.2 Historique

La figure IV.11 illustre les différentes chronologies de développement des métaheuristiques.

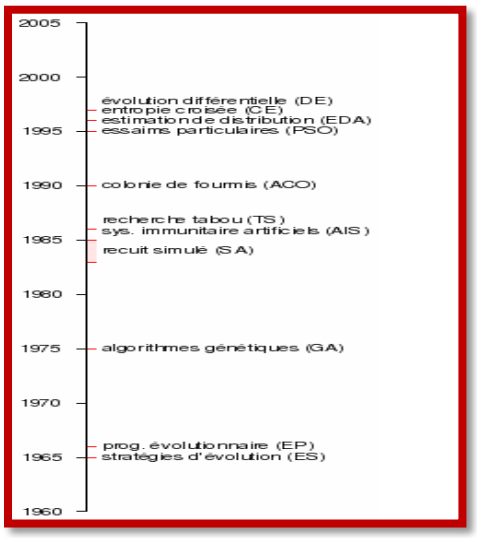

Figure IV.11 - Chronologie des principales métaheuristiques

Ici, on donne aussi les principales chronologies de développement des métaheuristiques.

1. Chronologie des principales métaheuristiques, le nom est indiqué suivi de l'accronyme anglais entre parenthèses.
2. 1952 : premiers travaux sur l'utilisation de méthodes stochastiques pour l'optimisation

3. 1954 : Barricelli effectue les premières simulations du processus d'évolution et les utilise sur des problèmes d'optimisation généraux
4. 1965 : Rechenberg conçoit le premier algorithme utilisant des stratégies d'évolution
5. 1966 : Fogel, Owens et Walsh proposent la programmation évolutionnaire
6. 1970 : Hastings propose l'algorithme de Metropolis-Hasting, permettant d'échantillonner n'importe-quelle distribution de probabilité
7. 1970 : John Horton Conway conçoit le jeu de la vie, l'automate cellulaire le plus connu à ce jour.
8. 1975 : travaillant sur les automates cellulaires, Holland propose les premiers algorithmes génétiques
9. 1980 : Smith utilise la programmation génétique
10. 1983 : s'appuyant sur les travaux d'Hastings, Kirkpatrick, Gelatt et Vecchi concoivent le recuit simulé
11. 1985 : indépendamment de ceux-ci, Černý propose le même algorithme
12. 1986 : La première mention du terme métaheuristique est faite par Fred Glover, lors de la conception de la recherche tabou : "La recherche avec tabou peut être vue comme une "métaheuristique", superposée à une autre heuristique. L'approche vise à éviter les optimums locaux par une stratégie d'interdiction (ou, plus généralement, de pénalisation) de certains mouvements."
13. 1986 : Farmer, Packard et Perelson travaillent sur les systèmes immunitaire artificiels.
14. 1988 : la première conférence sur les algorithmes génétiques est organisée à l'université de l'Illinois à Urbana-Champaign.
15. 1988 : des travaux sur le comportement collectif des fourmis trouvent une application en intelligence artificielle .
16. 1988 : Koza dépose son premier brevet sur la programmation génétique.
17. 1989 : Goldberg publie un des livres les plus connus sur les algorithmes génétiques.
18. 1989 : *Evolver*, le premier logiciel d'optimisation par algorithmes génétiques est publié par la société *Axcelis*.
19. 1989 : le terme algorithme mémétique apparait.
20. 1990 : Les algorithmes de colonie de fourmis sont proposés par Marco Dorigo, dans sa thèse de doctorat.
21. 1993 : le terme "Evolutionary Computation" (calcul évolutionnaire en français) se répand, avec la parution de la revue éponyme, publiée par le Massachusetts Institute of Technology.

22. 1995 : Feo et Resende proposent la méthode GRASP (pour Greedy randomized adaptive search procedure,procédure de recherche avide aléatoire adaptative" en français).
23. 1995 : Kennedy et Eberhart concoivent l'optimisation par essaims particulaires.
24. 1996 : Mühlenbein et Paaß proposent les algorithmes à estimation de distribution.
25. 1997 : Storn et Price proposent un algorithme à évolution différentielle.
26. 1997 : Rubinstein conçoit la méthode de l'entropie croisée.
27. 1999 : Boettcher propose l'optimisation extrémale.
28. 2000 : premiers algorithmes génétiques interactifs.

IV.5.3 Extensions

Les métaheuristiques, de part leur nature flexible, se prêtent particulièrement à des extensions. On peut ainsi trouver des adaptations pour des problèmes plus complexes que l'optimisation mono-objective (voir la figure IV.12) :

- problèmes multi-objectifs (plusieurs objectifs contradictoires doivent être optimisés de concert),
- optimisation multimodale (plusieurs optimums doivent être trouvés),
- optimisation en environnement incertain (un bruit est présent sur la valeur renvoyée par la fonction objective, ou sur le passage des paramètres à celle-ci),
- hybridations entre métaheuristiques,
- hybridations avec des méthodes exactes,
- problèmes dynamiques (la fonction objective change durant le temps de l'optimisation),
- gestion de contraintes fortement non linéaires,
- combinaison des problèmes précédents,
- etc.

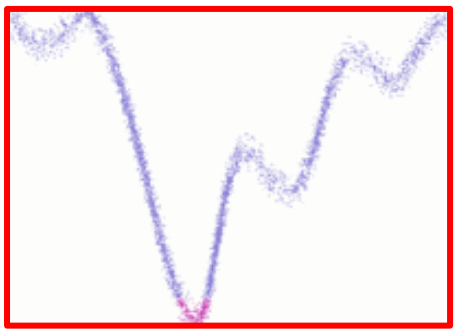

Figure IV.12- Exemple de problème d'optimisation continue, dynamique et bruité.

IV.6 – Références

1. (fr) Jacques Teghem et Marc Pilrot (éditeurs), Optimisation approchée en recherche opérationnelle, Hermes, traité I2C, mai 2002. ISBN 274620622
2. (fr) Yann Collette, Patrick Siarry, Optimisation multi-objectif, Éd. Eyrolles, Paris, 2002, Broché, 322 pages, ISBN 2-212-11168-1.
3. (fr) Johann Dréo, Alain Petrowski, Éric Taillard, Patrick Siarry, Métaheuristiques pour l'optimisation difficile, Français, Éd. Eyrolles, Paris, septembre 2003, Broché, 356 pages, ISBN 2-212-11368-4.
4. (en) Pierre Collet, Jean-Philippe Rennard, Introduction to Stochastic Optimization Algorithms, dans Handbook of Research on Nature-Inspired Computing for Economics and Management, édité par J.-P. Rennard, IDEA Group Inc, septembre 2006, Relié, 1100 pages, ISBN 1591409845.
5. (en) Christian Blum, Andrea Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys, volume 35, numéro 3, septembre 2003, pages 268-308. DOI:10.1145/937503.937505
6. V. Angel, La rugosité des paysages : une théorie pour la difficulté des problèmes d'optimisation combinatoire relativement aux métaheuristiques, thèse de doctorat de l'université de Paris-Sud, Orsay, 1998.
7. J. Dreo, J.-P. Aumasson, W. Tfaili, P. Siarry, Adaptive Learning Search, a new tool to help comprehending metaheuristics, to appear in the International Journal on Artificial Intelligence Tools.
8. El-Ghazali Talbi, A taxonomy of hybrid metaheuristics, Journal of Heuristics, volume 8, no 2, pages 541-564, septembre 2002
9. (en) exemples de fonctions de tests pour métaheuristiques d'optimisation de problèmes continus.
10. Robbins, H. and Monro, S., A Stochastic Approximation Method, Annals of Mathematical Statistics, vol. 22, pp. 400-407, 1951
11. Barricelli, Nils Aall, Esempi numerici di processi di evoluzione, Methodos, pp. 45-68, 1954
12. Rechenberg, I., Cybernetic Solution Path of an Expeimental Problem, Royal Aircraft Establishment Library Translation, 1965
13. Fogel, L., Owens, A.J., Walsh, M.J., Artificial Intelligence through Simulated Evolution, Wiley, 1966

14. W.K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications, Biometrika, volume 57, no 1, pages 97-109, 1970
15. Holland, John H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975
16. Smith, S.F., A Learning System Based on Genetic Adaptive Algorithms, PhD dissertation (University of Pittsburgh), 1980
17. S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, Optimization by Simulated Annealing, Science, volume 220, no 4598, pages 671-680, 1983
18. V. Cerny, A thermodynamical approach to the travelli
19. Fred GLover, Future Paths for Integer Programming and Links to Artificial Intelligence, Comput. & Ops. Res.Vol. 13, No.5, pp. 533-549, 1986
20. J.D. Farmer, N. Packard and A. Perelson, The immune system, adaptation and machine learning, Physica D, vol. 22, pp. 187--204, 1986
21. F. Moyson, B. Manderick, The collective behaviour of Ants : an Example of Self-Organization in Massive Parallelism, Actes de AAAI Spring Symposium on Parallel Models of Intelligence, Stanford, Californie, 1988
22. Koza, John R. Non-Linear Genetic Algorithms for Solving Problems. United States Patent 4,935,877. Filed May 20, 1988. Issued June 19, 1990
23. Goldberg, David E., Genetic Algorithms in Search, Optimization and Machine Learning, Kluwer Academic Publishers, Boston, MA., 1989
24. P. Moscato, On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms, Caltech Concurrent Computation Program, C3P Report 826, 1989.
25. M. Dorigo, Optimization, Learning and Natural Algorithms, Thèse de doctorat, Politecnico di Milano, Italie, 1992.
26. Feo, T., Resende, M., Greedy randomized adaptive search procedure, Jounral of Global Optimization, tome 42, page 32--37, 1992
27. Eberhart, R. C. et Kennedy, J., A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995
28. Kennedy, J. et Eberhart, R. C., Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995

29. Mülhenbein, H., Paaß, G., From recombination of genes to the estimation of distribution I. Binary parameters, Lectures Notes in Computer Science 1411: Parallel Problem Solving from Nature, tome PPSN IV, pages 178--187, 1996
30. Rainer Storn, Kenneth Price, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization, volume 11, no 4, pages 341-359, 1997
31. Rubinstein, R.Y., Optimization of Computer simulation Models with Rare Events, European Journal of Operations Research, 99, 89-112, 1997
32. Stefan Boettcher, Allon G. Percus, "Extremal Optimization: Methods derived from Co-Evolution", Proceedings of the Genetic and Evolutionary Computation Conference (1999)
33. Takagi, H., Active user intervention in an EC Search, Proceesings of the JCIS 2000
34. (en) Bibliographie d'introduction aux métaheuristiques

Sites généralistes :

1. (en) EU/ME the European chapter on metaheuristics (en) le metaheuristic network,

Logiciels et frameworks :

1. (en) HotFrame, un cadre généraliste pour la conception de métaheuristiques,
2. (en) Tempar, un cadre de conception pour les métaheuristiques combinatoires,
3. (en) Softwares for the design of parallel and hybrid metaheuristics, liste des logiciels tournants autour du projet "Evolving Objects".
4. (en) Open Metaheuristic, un cadre d'application et un logiciel libre pour la conception et les tests de métaheuristiques.