Métaheuristiques d'optimisation vues sous l'angle de l'échantillonnage de distribution

ARTICLE *in* JOURNAL EUROPÃ©EN DES SYSTÃ¨MES AUTOMATISÃ©S · MARCH 2008

DOI: 10.3166/jesa.42.9-30

2 AUTHORS:

Johann Dreo

Thales Research and Technology

41 PUBLICATIONS **486** CITATIONS

SEE PROFILE

Patrick Siarry

Université Paris-Est Créteil Val de Marne - Uni…

375 PUBLICATIONS **4,617** CITATIONS

SEE PROFILE

Métaheuristiques d'optimisation vues sous l'angle de l'échantillonnage de distribution

Johann Dréo[1], Patrick Siarry[2]

1. École Nationale Supérieure des Mines de Saint-Étienne
Centre ingénierie et santé
158 cours Fauriel, 42100 Saint-Étienne
dreo@emse.fr

2. Université de Paris XII Val-de-Marne
Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, EA 3956)
61, avenue du Général de Gaulle, 94010 Créteil
siarry@univ-paris12.fr

Résumé

Les métaheuristiques à population forment aujourd'hui la majorité des algorithmes utilisés pour des problèmes d'*optimisation difficile*. On présente souvent ces méthodes sous un angle purement algorithmique, en insistant sur les métaphores qui ont mené à leur conception. Nous proposons dans cet article de considérer les métaheuristiques à population comme des méthodes faisant évoluer un échantillonnage probabiliste de la fonction objectif via des processus d'apprentissage, de diversification et d'intensification. Après une description de quelques métaheuristiques, nous présentons une synthèse de leur fonctionnement vue sous cet angle. Nous concluons enfin sur l'idée que le coeur des différences entre métaheuristiques se situe au niveau de la façon d'échantillonner une distribution de probabilité, que ce soit explicitement, implicitement ou directement. La mise en avant de ces caractéristiques devrait permettre de faciliter la conception de nouvelles métaheuristiques.

1 Introduction

L'optimisation est un sujet central en recherche opérationnelle, un grand nombre de problèmes d'aide à la décision pouvant en effet être décrits sous la forme de problèmes d'optimisation. Les problèmes d'identification, d'apprentissage supervisé de réseaux de neurones ou encore la recherche du plus court chemin sont, par exemple, des problèmes d'optimisation.

Les métaheuristiques forment une famille d'algorithmes d'optimisation visant à résoudre approximativement des problèmes d'*optimisation difficile*, pour

lesquels on ne connaît pas de méthode classique plus efficace. Elles sont généralement utilisées comme des méthodes génériques pouvant traiter une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé.

Les métaheuristiques sont souvent classées selon deux ensembles : les algorithmes à base de solution courante unique et les méthodes à population. Dans le premier cas, la métaheuristique manipule un point, et décide à chaque itération quel sera le point suivant. On classe par exemple la recherche avec tabous et le recuit simulé dans cet ensemble. Dans le second cas, la métaheuristique manipule une population de points, un nouveau jeu de points est choisi à chaque itération. Beaucoup d'algorithmes peuvent entrer dans cet ensemble, comme les algorithmes évolutionnaires ou les algorithmes de colonies de fourmis.

Bien que ces classes soient perméables (un algorithme pouvant se trouver dans les deux classes, selon le point de vue où l'on se place), elles permettent de mettre en valeur une caractéristique importante : les métaheuristiques à population manipulent un ensemble de points, elles font évoluer un *échantillonnage* de la fonction objectif.

Cet article présente une revue de quelques métaheuristiques à population, parmi les plus connues, vues comme des algorithmes d'échantillonnage probabiliste d'une distribution. La section 2 présente les concepts de base que sont la programmation à mémoire adaptative et l'échantillonnage de la fonction objectif comme distribution de probabilité. Dans la section 3, nous présentons succinctement quelques métaheuristiques, en insistant sur ces aspects. La section 4 aborde quelques tentatives de mises en perspectives, ainsi qu'une proposition de synthèse des métaheuristiques présentées. Nous concluons l'article dans la section 5.

2 Concepts fondamentaux

Les métaheuristiques "à population" partagent un certain nombre de points communs. Elles manipulent toutes un échantillonnage de la fonction objectif, via des processus communs. Ces processus ont été rappelés dans le concept de programmation à mémoire adaptative, qui tente de synthétiser le coeur des métaheuristiques à population.

2.1 Fonction objectif et distribution de probabilité

Dans la majorité des métaheuristiques, l'échantillonnage est probabiliste, les meilleures solutions ayant idéalement plus de chances d'être sélectionnées. Cependant, dans un problème d'optimisation, le but n'est pas d'échantillonner la fonction objectif, mais de trouver le mode (l'optimum) de cette distribution. Ainsi, l'échantillonnage doit se concentrer sur les régions d'intérêt, en convergeant progressivement vers l'optimum. Du point de vue de l'échantillonnage, cette convergence s'effectue par une baisse progressive de la dispersion dans ces zones.

2.2 Programmation à mémoire adaptative et cadre I&D

Le concept de programmation à mémoire adaptative (*PMA*) [56, 57] est né de l'observation que les métaheuristiques récentes tendaient à devenir proches les unes des autres. Ainsi, du point de vue de la programmation à mémoire adaptative, certaines métaheuristiques partagent maintenant une démarche commune, présentée dans l'algorithme 1.

Algorithme 1 Démarche employée par un programme à mémoire adaptative.

1. Mémorisation d'un jeu de solutions ou une structure de données rassemblant les particularités des solutions produites par la recherche,
2. construction d'une solution provisoire sur la base des données mémorisées,
3. amélioration de la solution par un algorithme de recherche locale,
4. mémorisation de la nouvelle solution ou de la structure de données associée.

La programmation à mémoire adaptative insiste sur trois concepts fondamentaux : la *mémoire*, l'*intensification* et la *diversification*. Dans la littérature des algorithmes évolutionnaires, ces deux dernières notions sont souvent respectivement désignées par les termes *exploitation* et *exploration*, ayant un sens similaire. Nous avons choisi ici d'utiliser les termes de la définition originale de la programmation à mémoire adaptative, employés plus généralement dans la littérature de la recherche avec tabous ou du recuit simulé.

La mémoire représente ici l'information récoltée par l'algorithme, sur laquelle il va s'appuyer pour effectuer sa recherche. La mémoire est présente sous de nombreuses formes possibles, de la plus simple (une population de points) à des structures plus complexes (les pistes de phéromone des algorithmes de colonies de fourmis). La mémoire est une modélisation des solutions, elle est un moyen de décrire la fonction objectif. Cette modélisation s'opère par une distribution sur l'espace de recherche, biaisée vers les meilleures solutions, qui évolue en principe vers les optima du problème. Cette mémoire peut être définie comme globale (par rapport au problème dans son ensemble) ou inter-individuelle (d'une solution relativement à une autre).

L'intensification consiste en l'utilisation des informations disponibles pour améliorer la pertinence de celles-ci. Du point de vue des métaheuristiques, il s'agit dans les exemples extrêmes d'une recherche locale déterministe. Les algorithmes de recherche locale sont ainsi souvent employés en association avec d'autres métaheuristiques plus complexes, donnant lieu à des algorithmes *hybrides* [58]. On rencontre ainsi souvent l'algorithme du "simplexe" de Nelder-Mead [47], mais des métaheuristiques plus complexes, comme la recherche avec tabous, sont parfois employées.

La diversification est la recherche de nouvelles informations, afin d'augmenter la connaissance du problème. Elle fait souvent appel à des méthodes stochastiques, et il est pour le moment difficile de dégager des idées générales, tant la diversité d'approches de cette composante des métaheuristiques est grande.

En pratique, les trois composantes de la *PMA* sont liées, et il est parfois difficile de les repérer distinctement dans les métaheuristiques proposées [8]. De fait, les métaheuristiques tentent d'équilibrer la balance entre diversification et intensification, et bien souvent les améliorations d'une métaheuristique existante consistent à faire pencher la balance dans un sens ou dans l'autre.

Cet aspect a mené Blum et Roli à proposer le "cadre I&D", qui insiste également sur l'importance des notions d'intensification, de diversification et de mémoire. Le cadre I&D met en avant le fait que les différents composants d'une métaheuristique ne peuvent généralement pas être catégorisés comme opérateurs d'intensification ou de diversification au sens strict, mais peuvent être placés sur une échelle entre ces deux comportements extrêmes. Les auteurs ajoutent à cela une séparation entre les mécanismes utilisant une mémoire et ceux purement aléatoires. Le résultat est la possibilité de placer des composants entre trois pôles, selon qu'ils utilisent l'information provenant :

- de la fonction objectif,
- d'un processus aléatoire,
- d'autres fonctions (mémoire, apprentissage).

Dans le cadre I&D, chaque composant ainsi caractérisé peut, de plus, être considéré comme intrinsèque ou stratégique, selon qu'il est défini comme faisant partie intégrante de la métaheuristique ou comme sur-ajouté à sa structure de base. Cependant, ce cadre ne donne pas d'indications, autres que générales, sur la manière d'organiser les composants ou sur leurs relations avec les aspects d'échantillonnage.

Le concept de programmation à mémoire adaptative, quant à lui, se veut une forme de généralisation du *mode de fonctionnement* des métaheuristiques. Certaines métaheuristiques ont un principe qui semble d'emblée très proche de la *PMA*, c'est le cas par exemple de la méthode *GRASP* ("Greedy Randomized Adaptive Search Procedure") [52] ou encore des algorithmes à estimation de distribution (*EDA*) [41]. Cependant, la *PMA* propose une approche globale, sans entrer dans les détails de l'implémentation, qui induisent souvent des a priori sur la façon d'aborder tel ou tel problème.

2.3 Échantillonnage, PMA et I&D

On peut rapprocher la problématique de l'échantillonnage de distribution de la programmation à mémoire adaptative : l'échantillonnage de la fonction objectif est appelé diversification et la baisse de dispersion, intensification. La mémoire est présente dans le processus "d'apprentissage" des paramètres de la distribution manipulée.

Le point crucial dans cette perspective est en fait la description de la distribution de probabilité à utiliser. La plupart des algorithmes à population actuels n'ont aucun a priori sur la distribution et utilisent des techniques empiriques pour effectuer plus ou moins simultanément diversification et intensification. Cependant, certaines méthodes postulent que la fonction objectif peut être raisonnablement approchée par une distribution donnée, et vont donc utiliser cette distribution pour effectuer le tirage aléatoire de la population et la réduction

de dispersion. On peut parler de méthodes implicites dans le premier cas, et explicites dans le second [4].

Comme le suggère la PMA, les étapes de diversification permettent de rassembler de l'information sur le problème, ces informations sont mémorisées, puis triées par l'intensification, après quoi les informations en mémoire sont utilisées pour effectuer une nouvelle étape de diversification. Cette succession d'étapes apparaît clairement dès lors que l'on considère les métaheuristiques à population comme des algorithmes manipulant un échantillonnage. Elle reste cependant archétypale, comme le montre le cadre I&D, tant les processus de diversification et d'intensification sont complémentaires et liés.

3 Exemples de métaheuristiques vues sous l'angle de l'échantillonnage de distribution

Nous verrons dans cette section comment quelques métaheuristiques emploient de fait l'échantillonnage de distribution pour résoudre un problème d'optimisation.

3.1 Recuit simulé

Le recuit simulé est fondé sur une analogie entre un processus physique (le recuit) et le problème de l'optimisation. Le recuit simulé en tant que métaheuristique [39, 13] s'appuie en effet sur des travaux visant à simuler l'évolution d'un solide vers son état d'énergie minimale [43, 35].

La description classique du recuit simulé le présente comme un algorithme probabiliste, où un point évolue sur l'espace de recherche. En effet, la méthode repose sur l'algorithme de Metropolis (ou méthode de Metropolis-Hastings, du nom de l'auteur de la généralisation de la méthode), présenté sur la figure 2, qui décrit un processus Markovien [1, 40] dont le but est d'échantillonner n'importe quelle distribution de probabilité. Le recuit simulé (dans sa version la plus courante, dite "homogène") appelle à chaque itération cette méthode.

Cependant, il est possible de voir le recuit simulé comme un algorithme à population [14]. En effet, l'algorithme de Metropolis est une méthode d'échantillonnage d'une distribution de probabilité : il échantillonne directement la fonction objectif par le biais d'une distribution de Bolzmann paramétrique (de paramètre T). Un des paramètres cruciaux est donc la décroissance de la température ; de nombreuses lois de décroissance différentes ont été proposées [59]. Il existe par ailleurs des versions du recuit simulé qui mettent en avant la manipulation d'une population de points [37, 60, 42, 38].

Ici, la méthode de Metropolis (ou toute autre méthode d'échantillonnage [16, 49]) tient lieu de diversification, associée à la décroissance de température, qui contrôle l'intensification.

Algorithme 2 La méthode de Metropolis permet de produire un échantillon de points suivant une distribution de probabilité dont on sait uniquement calculer la densité en un point x. Le taux d'acceptation (la fraction des points x_i proposés qui est acceptée) est lié à la "température" T dans le cas du recuit simulé.

Initialiser un point de départ x_0 et une *température* T

Pour $i = 1$ à n :

  Jusqu'à x_i accepté

    Si $f(x_i) \leq f(x_{i-1})$: accepter x_i

    Si $f(x_i) > f(x_{i-1})$: accepter x_i avec la probabilité $e^{-\frac{f(x_i)-f(x_{i-1})}{T}}$

  Fin

Fin

3.2 Algorithmes évolutionnaires

Les algorithmes évolutionnaires sont inspirés de l'évolution biologique des êtres vivants, qui décrit comment des espèces s'adaptent à leur environnement. L'analogie avec un problème d'optimisation [29] a donné lieu à plusieurs approches [36, 28, 51], parmi lesquelles les algorithmes génétiques sont sans doute l'exemple le plus connu [31].

Les algorithmes évolutionnaires manipulent une *population* d'*individus* : un ensemble de points dans l'espace de recherche. Chaque individu est lié à une valeur de la fonction objectif du problème, dénommée *fitness*, qui représente son degré d'adaptation. Les algorithmes évolutionnaires font évoluer cette population d'individus par *générations* successives, en utilisant des *opérateurs* inspirés de la théorie de l'évolution :

la sélection, qui permet aux individus les mieux adaptés de se reproduire le plus souvent ;

le croisement, qui produit un nouvel individu à partir de deux parents, en *recombinant* les caractéristiques de ceux-ci ;

la mutation, qui fait varier les caractéristiques d'un seul individu de façon aléatoire.

Beaucoup d'algorithmes évolutionnaires s'appuient également sur la notion de *représentation* des individus. Les individus sont ainsi classiquement représentés par des *chromosomes*, qui forment une liste d'entiers, un vecteur de nombres réels, ... Ce sont ces chromosomes qui sont modifiés par les opérateurs précédents.

En pratique, chaque itération de l'algorithme représente une génération. L'algorithme effectue ainsi une première phase de sélection, où sont désignés les individus qui vont participer à la phase suivante de croisement, puis de mutation. La dernière phase évalue la performance des individus, avant de passer à la prochaine génération.

Algorithme 3 La métaheuristique PBIL construit à chaque itération un ensemble de m solutions x, à l'aide d'un vecteur de probabilité $p(x)$. La mise à jour de $p(x)$ est guidée par un coefficient d'apprentissage α.

Initialiser un vecteur de probabilité $p_0(x)$

Jusqu'à critère d'arrêt :

Construire m individus $x_1^l, \ldots, x_m^l$ en utilisant $p_l(x)$

Évaluer et trier $x_1^l, \ldots, x_m^l$

Mémoriser les k meilleurs individus $x_{1:k}^l, \ldots, x_{m:k}^l$

Reconstruire le vecteur de probabilité :

$p_{l+1}(x) = (p_{l+1}(x_1), \ldots, p_{l+1}(x_k))$

Pour $i = 1$ à n :

$p_{l+1}(x_i) = (1-\alpha)\, p_l(x_i) + \alpha \frac{1}{k} \sum_{j=1}^{k} x_{i,j:k}^l$

Fin

Fin

Les algorithmes évolutionnaires classiques sont implicites. On peut cependant observer que les opérateurs de croisement et de mutation visent à produire un ensemble de nouveaux points (diversification) dans les limites de la population précédente (mémoire), points dont on va ensuite réduire le nombre (intensification), et ainsi de suite.

La difficulté à manipuler les opérateurs de diversification a entraîné la création de méthodes explicites (voir notamment [55], [34, 33], [6]), qui démontrent plus clairement le rapport des algorithmes évolutionnaires avec la notion d'échantillonnage de distribution. La sous-section suivante décrit succinctement l'une de ces méthodes.

La méthode *PBIL* L'algorithme *PBIL* ("Population-Based Incremental Learning") est à l'origine inspiré de l'apprentissage compétitif et est conçu pour des problèmes binaires. Il fait l'objet d'une large littérature, on se référera notamment à [2, 5, 3], ainsi qu'à [53] pour le domaine continu.

La méthode *PBIL*, décrite sur l'algorithme 3, utilise un coefficient d'apprentissage (noté ici α) qui contrôle l'amplitude des changements. La version continue utilise une distribution gaussienne à base d'un produit de densités univariantes et indépendantes, ainsi que plusieurs adaptations pour l'évolution du vecteur de variance.

L'algorithme enchaîne de façon itérative des étapes de diversification, de mémorisation et d'intensification sur un vecteur de probabilité (pour la version combinatoire) : nous sommes bien en présence d'un échantillonnage probabiliste.

3.3 Algorithmes à estimation de distribution

Les algorithmes à estimation de distribution ("Estimation of Distribution Algorithms", *EDA*) ont été conçus à l'origine comme une variante des algorithmes évolutionnaires [46]. Cependant, dans les méthodes de type *EDA*, il n'y a pas d'opérateurs de croisement ou de mutation. En effet, la population des nouveaux individus est tirée au hasard, selon une distribution estimée depuis des informations issues de la population précédente. Dans les algorithmes évolutionnaires, la relation entre les différentes variables est implicite alors que, dans les algorithmes *EDA*, le coeur de la méthode consiste justement à estimer ces relations, à travers l'estimation de la distribution de probabilité associée aux individus sélectionnés.

La meilleure manière de comprendre les *EDA* est d'étudier un exemple le plus simple possible. Ici, prenons comme problème la fonction cherchant à maximiser le nombre de 1 sur trois dimensions : on cherche donc à maximiser $h(x) = \sum_{i=1}^{3} x_i$ avec $x_i = \{0, 1\}$ (problème "OneMax").

La première étape consiste à engendrer la population initiale, on tire donc au hasard M individus, selon la distribution de probabilité : $p_0(x) = \prod_{i=1}^{3} p_0(x_i)$, où la probabilité que chaque élément x_i soit égal à 1 vaut $p_0(x_i)$. En d'autres termes, la distribution de probabilité, à partir de laquelle le tirage est fait, est factorisée comme un produit des trois distributions de probabilité marginales univariantes (ici, puisque les variables sont binaires, des distributions de Bernouilli de paramètre 0.5). La population ainsi échantillonnée est nommée D_0. Prenons pour l'exemple une population de six individus (illustrée figure 1).

i	x_1	x_2	x_3	$h(x)$
1	1	1	0	2
2	0	1	1	2
3	1	0	0	1
4	1	0	1	2
5	0	0	1	1
6	0	1	0	1

FIG. 1 – L'algorithme EDA optimisant le problème OneMax : la population initiale D_0.

La seconde étape consiste à sélectionner des individus parmi cette population ; on construit ainsi de façon probabiliste une deuxième population $D_0^{S_e}$, tirée parmi les meilleurs individus de D_0. La méthode de sélection est libre. Ici, on peut, pour l'exemple, sélectionner les trois meilleurs individus (figure 2).

La troisième étape consiste à estimer les paramètres de la distribution de probabilité représentée par ces individus sélectionnés. Dans cet exemple, on considère que les variables sont indépendantes ; trois paramètres permettent donc de caractériser la distribution. On va donc estimer chaque paramètre $p(x_i \mid D_0^{S_e})$ par sa fréquence relative dans $D_0^{S_e}$. On a donc : $p_1(x) = p_1(x_1, x_2, x_3) = \prod_{i=1}^{3} p(x_i \mid D_0^{S_e})$. Ici, la probabilité $p(x)$ de trouver un 1 dans chaque x_i est de

i	x_1	x_2	x_3	$h(x)$
1	1	1	0	2
2	0	1	1	2
4	1	0	1	2
$p(x)$	0.66	0.66	0.66	

FIG. 2 – L'algorithme EDA optimisant le problème OneMax : les individus sélectionnés $D_0^{S_e}$.

0.66. En échantillonnant cette distribution de probabilité $p_1(x)$, on peut obtenir une nouvelle population D_1 (figure 3).

i	x_1	x_2	x_3	$h(x)$
1	1	1	1	3
2	1	1	1	3
3	1	0	0	1
4	1	0	1	2
5	0	1	0	1
6	1	1	0	2

FIG. 3 – L'algorithme EDA optimisant le problème OneMax : la nouvelle population D_1.

On continue ainsi jusqu'à ce que l'on atteigne un critère d'arrêt prédéterminé... L'algorithme général d'une méthode à estimation de distribution est présenté sur l'algorithme 4.

La principale difficulté, lorsqu'on utilise un algorithme à estimation de distribution, est d'estimer la distribution de probabilité. En pratique, il faut approcher les paramètres de la distribution, conformément à un modèle choisi. De nombreuses approximations ont été proposées pour des problèmes d'optimisation continue, aussi bien que combinatoire. On peut classer les différents algorithmes proposés selon la complexité du modèle utilisé pour évaluer les dépendances entre variables. On définit ainsi trois catégories :

1. Modèles *sans* dépendance : la distribution de probabilité est factorisée à partir de distributions indépendantes univariantes pour chaque dimension. Ce choix a le défaut d'être peu réaliste dans les cas de l'optimisation difficile, où une dépendance des variables est généralement la règle ;
2. Modèles à dépendances *bivariantes* : la distribution de probabilité est factorisée à partir de distributions bivariantes. Dans ce cas, l'apprentissage de la distribution peut être étendu jusqu'à la notion de *structure* ;
3. Modèles à dépendances *multiples* : la factorisation de la distribution de probabilité est obtenue à partir de statistiques d'ordre *supérieur* à deux.

On peut noter que, dans le cas de problèmes continus, le modèle de distribution est souvent fondé sur une base de distribution normale.

Algorithme 4 Algorithme à estimation de distribution.

$D_0 \leftarrow$ Engendrer M individus aléatoirement

$i = 0$

Tant que critère d'arrêt :

$i = i + 1$

$D_{i-1}^{S_e} \leftarrow$ Sélectionner $N \leq M$ individus dans D_{i-1} grâce à la méthode de sélection

$p_i(x) = p\left(x \mid D_{i-1}^{S_e}\right) \leftarrow$ Estimer la distribution de probabilité des individus sélectionnés

$D_i \leftarrow$ Échantillonner M individus depuis $p_i(x)$

Fin

Quelques variantes d'importance ont été proposées : l'utilisation de "data clustering" pour l'optimisation multimodale et des variantes parallèles pour des problèmes combinatoires. Des théorèmes de convergence ont également été formulés, notamment avec l'aide de modélisations par chaînes de Markov ou par systèmes dynamiques.

Il serait fastidieux d'énumérer ici les algorithmes proposés dans chaque cas, on se reportera pour cela au très complet livre de référence dans le domaine [41].

Dans la grande majorité des EDA, l'accent est porté sur le fait que la diversification utilise des distributions de probabilité explicites, l'intensification n'étant, quant à elle, effectuée que par un opérateur de sélection.

3.4 Algorithmes de colonies de fourmis

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées *phéromones*. Elles sont très sensibles à ces substances, qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. Ces substances sont nombreuses et varient selon les espèces. Les fourmis peuvent déposer des phéromones au sol, grâce à une glande située dans leur abdomen, et former ainsi des pistes odorantes, qui pourront être suivies par leurs congénères.

Les fourmis utilisent les pistes de phéromone pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Une colonie est ainsi capable de choisir (sous certaines conditions) le plus court chemin vers une source à exploiter [32, 7], sans que les individus aient une vision *globale* du trajet.

Le premier algorithme de colonies de fourmis [15] a été conçu pour résoudre le problème du voyageur de commerce. Ce problème consiste à chercher le trajet le plus court reliant n villes données, chaque ville ne devant être visitée qu'une seule fois.

Dans l'algorithme du "Ant System" (AS) [15], à chaque itération, chaque fourmi parcourt le graphe et construit un trajet complet de n étapes. Pour chaque fourmi, le trajet entre une ville i et une ville j dépend de :

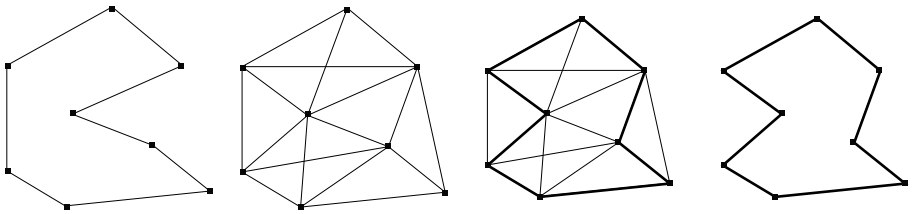


FIG. 4 – Le problème du voyageur de commerce optimisé par l'algorithme AS, les points représentent les villes et l'épaisseur des arêtes la quantité de phéromone déposée. (a) exemple de trajet construit par une fourmi, (b) au début du calcul, tous les chemins sont explorés, (c) le chemin le plus court est davantage renforcé que les autres, (d) l'évaporation permet d'éliminer les moins bonnes solutions.

1. la liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi k est sur la ville i : J_i^k,
2. l'inverse de la distance entre les villes : $\eta_{ij} = \frac{1}{d_{ij}}$, appelé *visibilité*. Cette information statique est utilisée pour diriger le choix des fourmis vers des villes proches,
3. la quantité de phéromone déposée sur l'arête reliant les deux villes, appelée l'*intensité de la piste*. Ce paramètre définit l'attractivité d'une partie du trajet global et change à chaque passage d'une fourmi. C'est, en quelque sorte, une mémoire globale du système, qui évolue par apprentissage.

La règle de déplacement (appelée "règle aléatoire de transition proportionnelle" par les auteurs [9]) contrôle l'importance relative de l'*intensité* de la piste et de la *visibilité*.

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromone sur l'ensemble de son parcours, quantité qui dépend de la *qualité* de la solution trouvée.

L'algorithme ne serait pas complet sans le processus d'*évaporation* des pistes de phéromone. En effet, pour éviter d'être piégé dans des solutions sous-optimales, il est nécessaire de permettre au système "d'oublier" les mauvaises solutions. On contrebalance donc l'additivité des pistes par une décroissance constante des valeurs des arêtes à chaque itération.

La figure 4 présente un exemple simplifié de problème du voyageur de commerce optimisé par un algorithme *AS*.

Les algorithmes de colonies de fourmis sont pour l'essentiel appliqués à des problèmes combinatoires, notamment du type du voyageur de commerce [30, 21, 20, 19]. Cependant, devant le succès rencontré par ces algorithmes, d'autres pistes commencent à être explorées : par exemple, l'utilisation de ces algorithmes dans des problèmes *continus* [26] et/ou *dynamiques* [18], ou encore l'exploitation de ce type d'algorithmes dans un cadre d'*intelligence en essaim* [9] et avec d'autres métaheuristiques [44, 45, 24, 25]. Le livre de référence sur

l'intelligence en essaim [9] insiste sur les aspects biologiques du domaine et présente un grand nombre d'algorithmes. Un autre livre, plus récent, se restreint aux algorithmes de colonies de fourmis [22].

Du point de vue de l'estimation de distribution, le point clef concerne le choix des composants de la solution. En effet, du fait des pistes de phéromone, chaque fourmi choisit de façon *probabiliste* un composant parmi d'autres. Ainsi, chaque solution est associée à une certaine probabilité de choix, probabilité résultant de l'ensemble des probabilités des composants, mises à jour de façon constructive.

Très récemment, des approches facilement comparables à des problèmes d'échantillonnage ont été employées dans la conception d'une métaheuristique du type colonie de fourmis pour des problèmes continus [54, 50]. Dans les deux cas, le principe de l'algorithme est comparable. Dans ces méthodes, la distribution de probabilité mémorisée sous la forme de "phéromones" τ est celle qui est échantillonnée et dont l'évolution va guider la recherche. Dans les deux cas, il s'agit en théorie d'une somme de distributions normales univariées.

Si l'on observe les algorithmes de colonies de fourmis avec une approche "haut niveau", il y a bien échantillonnage et manipulation d'une distribution de probabilité [25, 23], qui va tendre à évoluer vers un optimum. Dans les modèles continus, la diversification est faite par échantillonnage d'une distribution à base normale, et l'intensification se fait par sélection des meilleurs individus, appelée ici "évaporation".

4 Synthèse

4.1 Cadres généraux

Suite aux origines indépendantes de métaheuristiques se révélant finalement très proches, plusieurs tentatives de structuration dans le sens de l'échantillonnage de distribution ont vu le jour.

Monmarché *et al.* proposent par exemple un modèle appelé *PSM* ("Probabilistic Search Metaheuristic" [44, 45]), en se fondant sur la comparaison des algorithmes *PBIL* ([2, 5], décrits dans le paragraphe 3.2), *BSC* [55] et du *Ant System* ([15], décrit dans la section 3.4). Le principe général d'une méthode *PSM* est présenté sur l'algorithme 5. On peut constater la parenté de cette approche avec les algorithmes à estimation de distribution, mais l'approche *PSM* se restreint à l'utilisation de vecteurs de probabilité, en précisant toutefois que la règle de mise à jour de ces vecteurs est cruciale.

Les *EDA* ont été présentés dès le départ comme des algorithmes évolutionnaires où la diversification serait explicite [46]. Ce sont sans doute les algorithmes les plus proches d'un cadre général. Une généralisation de ces méthodes aux cas continus et discrets a été proposée sous le terme de *IDEA* ("Iterated Density Evolutionary Algorithms", [10, 11, 12]), elle se dote notamment de bases mathématiques, dont les auteurs de *PSM* regrettaient l'absence dans ce genre d'approche [44]. Le principe de l'approche *IDEA* est présenté sur l'algorithme 6.

Algorithme 5 Le cadre de la méthode *PSM*.

Initialiser un vecteur de probabilité $p_0(x)$

Jusqu'à critère d'arrêt :

Construire m individus $x_1^l, \ldots, x_m^l$ en utilisant $p_l(x)$

Évaluer $f\left(x_1^l\right), \ldots, f\left(x_m^l\right)$

Reconstruire le vecteur de probabilité $p_{l+1}(x)$ en tenant compte de $x_1^l, \ldots, x_m^l$ et de $f\left(x_1^l\right), \ldots, f\left(x_m^l\right)$

Fin

Algorithme 6 L'approche *IDEA*.

Initialiser une population P_0 de n points

Tant_que critère d'arrêt non atteint :

Mémoriser le plus mauvais point θ

Chercher une distribution $D_i(X)$ adéquate à partir de la population P_{i-1}

Construire une population O_i de m points selon $D_i(X)$, avec $\forall O_i^j \in O_i$: $f\left(O_i^j\right) < f(\theta)$

Créer une population P_i à partir d'une partie de P_{i-1} et d'une partie de O_i

Évaluer P_i

Fin

Algorithme 7 Synthèse de la méthode du recuit simulé.

Initialisation d'un ensemble de points

Itérations juqu'au critère d'arrêt

Apprentissage : [mécanismes de relation entre l'ensemble de points et la température]

Diversification : échantillonnage *direct* de la fonction objectif par la méthode de Metropolis

Intensification : baisse de la température

Fin

IDEA utilise une diversification plus générale que *PSM*, en ne se limitant pas à un vecteur de probabilité comme modèle, mais en précisant que la recherche de la meilleure distribution de probabilité fait partie intégrante de l'algorithme. Cependant, la baisse de dispersion s'effectue en sélectionnant les meilleurs individus, aucune précision sur l'utilisation de principes d'intensification différents n'est donnée.

4.2 Résumé

Afin de pouvoir comparer rapidement les algorithmes que nous venons de présenter dans le cadre de l'échantillonnage de distribution, nous avons tenté de résumer chaque métaheuristique. Nous avons mis en avant les trois phases de la programmation à mémoire adaptative : apprentissage (mémorisation), diversification et intensification. Pour chaque algorithme, il est nécessaire de garder à l'esprit que la classification proposée reste plus ou moins arbitraire. Ainsi, une phase présentée comme étant une étape de diversification contient vraisemblablement un aspect d'intensification ou d'apprentissage. Nous avons néanmoins choisi de simplifier la présentation, en ne considérant que l'aspect majoritaire.

L'algorithme 7 présente tout d'abord la synthèse du recuit simulé. L'étape d'apprentissage n'est pas présente dans les versions de base, mais il existe cependant certaines variantes qui tentent de relier la température à certaines caractéristiques de l'échantillonnage effectué par la méthode de Metropolis [27, 48, 17]. Au final, le recuit simulé se caractérise surtout par le fait qu'il échantillonne directement la fonction objectif.

L'algorithme 8 présente une synthèse des algorithmes évolutionnaires. Chaque étape peut ici être facilement reliée à l'un des principaux opérateurs. Les algorithmes évolutionnaires n'ont cependant aucun a priori sur la distribution de probabilité utilisée et n'utilisent pas la fonction objectif directement. De fait, il s'agit avant tout d'une méthode empirique, où l'on peut aisément ajouter un ou plusieurs opérateurs pour orienter la métaheuristique vers un comportement voulu, ou l'adapter à un problème donné.

Algorithme 8 Synthèse des algorithmes évolutionnaires.

Initialisation d'un ensemble de points

Itérations juqu'au critère d'arrêt

Apprentissage : croisement

Diversification : mutation

Intensification : sélection

Fin

Algorithme 9 Synthèse des algorithmes de colonies de fourmis.

Initialisation d'un ensemble de points

Itérations juqu'au critère d'arrêt

Apprentissage : dépôt de pistes de phéromones et évaporation

Diversification : influence de la visibilité

Intensification : influence des pistes de phéromones

Fin

L'algorithme 9 présente la synthèse des algorithmes de colonies de fourmis. Ces méthodes probabilistes effectuent un apprentissage par le biais des pistes de phéromones, qui forment une certaine description du problème. La particularité de ces algorithmes apparaît sur les problèmes combinatoires, où ils construisent itérativement les solutions, sans les manipuler une à une.

L'algorithme 10 présente la synthèse des algorithmes à estimation de distribution. Il s'agit sans doute ici de la méthode la plus évidente à appréhender dans cette optique. Ces métaheuristiques utilisent une distribution de probabilité explicitement déterminée comme base d'échantillonnage.

Algorithme 10 Synthèse des algorithmes à estimation de distribution.

Initialisation d'un ensemble de points

Itérations juqu'au critère d'arrêt

Apprentissage : extraction des paramètres d'une distribution *explicite*

Diversification : échantillonnage de la distribution

Intensification : sélection

Fin

4.3 Recherche à apprentissage adaptatif

Afin de clarifier le mode de fonctionnement des métaheuristiques, nous proposons de combiner les idées en provenance de la PMA, du cadre I&D et de la notion d'échantillonnage de distribution au sein du concept de "recherche à apprentissage adaptatif" (RAA). Dans ce cadre, les opérateurs d'intensification sont décrits comme permettant une réduction de la dispersion de l'échantillonnage de la fonction objectif et les opérateurs de diversification comme permettant la construction de cet échantillonnage. Les opérateurs d'apprentissage agissent, quant à eux, sur la distribution de probabilité utilisée par les opérateurs précédents. Entre ces trois extrêmes existe un large panel de variantes combinant les différents aspects. Les archétypes de ces comportements sont les suivants :

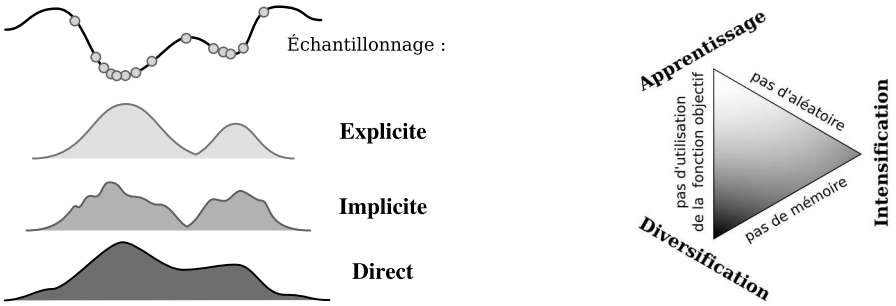
- intensification : l'échantillonnage s'appuie uniquement sur la fonction objectif (recherche locale) ;
- diversification : l'échantillonnage est purement aléatoire (mutation uniforme, bruitage, etc.) ;
- apprentissage : utilisation d'une distribution produite à partir de l'ensemble de l'échantillonnage effectué depuis le début du déroulement de l'algorithme.

Dans le cadre de la RAA, il est possible de répartir les métaheuristiques en trois catégories, selon le type d'échantillonnage utilisé :

- utilisation d'une description implicite de la distribution utilisée pour effectuer l'échantillonnage (méthodes évolutionnaires classiques, algorithmes de colonies de fourmis, plus généralement métaheuristiques à population dites "classiques"),
- utilisation d'une description explicite (méthodes évolutionnaires explicites et plus généralement algorithmes à estimation de distribution),
- utilisation directe de la fonction objectif (recuit simulé).

Les méthodes implicites ont l'avantage de pouvoir s'affranchir du choix d'une description de l'échantillonnage à utiliser, mais aussi le défaut d'être difficiles à manipuler et à comprendre. Les méthodes explicites permettent de maîtriser complètement les processus de diversification et d'intensification de façon indépendante, mais sont liées au choix d'une distribution de probabilité. Les méthodes directes permettent d'utiliser la distribution de probabilité "idéale" (la fonction objectif), mais rendent l'intensification délicate, car indépendante de la structure du problème.

En considérant les opérateurs d'échantillonnage conjointement avec les composants situés dans l'ensemble intensification-diversification-apprentissage, il devient plus aisé de combiner les processus de différentes métaheuristiques pour concevoir de nouvelles méthodes. Cette approche, appelée *hybridation de bas niveau* [58] ou *échange de composants* [8] est une des plus employées, mais se limite généralement à l'inclusion de méthodes de parcours au sein de métaheuristiques à population, ou à l'emploi de recherches locales ou déterministes. Nous proposons de considérer des hybridations plus générales, où il ne s'agirait plus d'intégrer des composants spécialisés dans une métaheuristique (l'aspect



(a) Types d'échantillonnages (b) Catégories de composants

FIG. 5 – Les trois classes de métaheuristiques proposées dans le cadre de la recherche à apprentissage adaptatif sont définies selon la manière de procéder à l'échantillonnage de la fonction objectif. Elles se combinent avec les trois archétypes de composants, qui déterminent un ensemble de comportements envisageables.

"stratégique" pointé dans le cadre I&D), mais bien de mélanger des opérateurs généraux entre eux.

Ainsi, la structure de plus haut niveau n'est plus la métaphore de la métaheuristique (algorithme "génétique" ou "recuit simulé", par exemple), mais le type d'échantillonnage ("implicite" ou "direct"), et les composants ne sont plus des aspects de la métaheuristique ("mutation" ou "dépôt de phéromones"), mais des opérateurs situés dans l'ensemble intensification-diversification-apprentissage.

5 Conclusion

Dans ce travail, nous avons présenté quelques-unes des métaheuristiques les plus connues, afin de montrer que les métaheuristiques à population pouvaient être vues comme des algorithmes manipulant un échantillonnage de la fonction objectif. Ces considérations ont donné lieu à la proposition du cadre de *recherche à apprentissage adaptatif* (RAA).

Dans la RAA, les métaheuristiques peuvent être décrites comme la combinaison d'opérateurs manipulant un échantillonnage produit sur la base d'une distribution de probabilité. Au plus haut niveau, les manières de construire la distribution de probabilité peuvent se répartir en trois classes : explicite (utilisation d'un modèle de distribution connue), implicite (modèle inconnu) et direct (la fonction objectif est le modèle de la distribution utilisée). Les différents composants des métaheuristiques permettant de faire évoluer l'échantillonnage entre deux itérations peuvent être répartis selon trois comportements archétypaux : intensification (réduction de la dispersion de l'échantillonnage, fondée uniquement sur la fonction objectif), diversification (tirage aléatoire dans une

distribution donnée) et apprentissage (utilisation de mémoire pour biaiser la distribution de probabilité).

L'intérêt de considérer les métaheuristiques sous cette forme est de faciliter la compréhension de leur fonctionnement, mais aussi et surtout la conception de nouveaux algorithmes. Une des perspectives le plus souvent citée comme intéressante est en effet l'intégration d'idées en provenance de diverses métaheuristiques, généralement sous la forme d'hybridations. Le cadre de la recherche à apprentissage adaptatif permet également d'éviter les problèmes de vocabulaires liés aux métaphores, très souvent employées, et il permet de faciliter les rapprochements avec d'autres disciplines.

Ainsi, les liens entre les métaheuristiques et l'apprentissage artificiel apparaissent plus clairement, montrant l'intérêt d'une utilisation de méthodes d'estimation de distributions de probabilité au sein d'algorithmes d'optimisation. D'autres perspectives s'ouvrent également, dans le domaine de l'hybridation entre métaheuristiques, où la levée du frein posé par la présentation des algorithmes sous forme de métaphores devrait permettre la construction de méthodes issues de la combinaison des composants les plus efficaces.

Références

[1] E. H. L. Aarts and P. J. M. Van Laarhoven. Statistical cooling : a general approach to combinatorial optimisation problems. *Philips J. of Research*, 40 :193–226, 1985.

[2] S. Baluja. Population-based Incremental Learning : A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.

[3] S. Baluja. An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University, 1995.

[4] S. Baluja. Genetic Algorithms and Explicit Search Statistics. *Advances in Neural Information Processing Systems*, 9 :319–325, 1997.

[5] S. Baluja and R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. In A. Prieditis and S. Russel, editors, *International Conference on Machine Learning*, pages 38–46, Lake Tahoe, California, 1995. Morgan Kaufmann.

[6] S. Baluja and S. Davies. Fast probabilistic modeling for combinatorial optimization. In *Fifteenth National Conference on Artificial Intelligence, Tenth Conference on Innovative Applications of Artificial Intelligence*, Madison, Wisconsin, 1998.

[7] R. Beckers, J. L. Deneubourg, and S. Goss. Trails and U-Turns in the Selection of a Path by the Ant Lasius Niger. *J. Theor. Biol.*, 159 :397–415, 1992.

[8] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Computing Surveys*, 35(3) :268–308, 2003.

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence, From Natural to Artificial Systems*. Oxford University Press, 1999.

[10] P. A. N. Bosman and D. Thierens. An algorithmic framework for density estimation based evolutionary algorithm. Technical Report UU-CS-1999-46, Utrech University, 1999.

[11] P.A.N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In M. Muehlenbein and A.O. Rodriguez, editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference GECCO-2000*, pages 197–200, San Francisco, California, 2000. Morgan Kauffmann.

[12] P.A.N. Bosman and D. Thierens. IDEAs based on the normal kernels probability density function. Technical Report UU-CS-2000-11, Utrecht University, 2000.

[13] V. Cerny. Thermodynamical approach to the traveling salesman problem : an efficient simulation algorithm. *J. of Optimization Theory and Applications*, 45(1) :41–51, 1985.

[14] Y. Collette. Les nouvelles variantes du recuit simulé. In *Journée META-SCDD*, Créteil, France, 1 April 2004. http ://lass.univ-lyon1.fr/gtscdd/journee01avr/Transp_Collette_01_04_04.ppt.

[15] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In F. Varela and P. Bourgine, editors, *Proceedings of ECAL'91 - First European Conference on Artificial Life*, pages 134–142, Paris, France, 1992. Elsevier Publishing.

[16] M. Creutz. Microcanonical Monte Carlo simulation. *Physical Review Letters*, 50(19) :1411–1414, May 1983.

[17] P. M. C. De Oliveira. Broad Histogram : An Overview. *Brazilian Journal of Physics*, 30, 2000.

[18] G. Di Caro and M. Dorigo. AntNet : Distributed stigmergic control for communications networks. *Journal of Artificial Intelligence Research*, 9 :317–365, 1998.

[19] M. Dorigo and L. M. Gambardella. Ant Colonies for the Traveling Salesman Problem. *BioSystems*, 43 :73–81, 1997.

[20] M. Dorigo and L. M. Gambardella. Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. Evol. Comp.*, 1 :53–66, 1997.

[21] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System : Optimization by a Colony of Cooperating Agents. *IEEE Trans. Syst. Man Cybern*, B(26) :29–41, 1996.

[22] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[23] J. Dréo. *Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical*. PhD thesis, Université Paris 12 Val de Marne, 13 December 2004.

[24] J. Dréo and P. Siarry. Diverses techniques d'optimisation inspirées de la théorie de l'auto-organisation dans les systèmes biologiques. In *Journée optimisation par essaim particulaire (OEP'2003)*, 2 October 2003.

[25] J. Dréo and P. Siarry. Algorithmes à estimation de distribution et colonies de fourmis. In *11ème journée évolutionnaire (JET11)*, 12 March 2004.

[26] J. Dréo and P. Siarry. Continuous Interacting Ant Colony Algorithm Based on Dense Heterarchy. *Future Generation Computer Systems*, 20(5) :841–856, 2004.

[27] A. M. Ferrenberg and R. H. Swendsen. Optimized Monte Carlo Data Analysis. *Phys. Rev. Lett.*, 63 :1195, 1989.

[28] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artifical Intelligence through Simulated Evolution*. Wiley, 1966.

[29] Alex S. Fraser. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Sciences*, 10 :484–491, 1957.

[30] L. M. Gambardella and M. Dorigo. Ant-Q : A Reinforcement Learning Approach to the Travelling Salesman Problem. In *Proceedings Twelfth International Conference on Machine Learning*, volume ML-95, pages 252–260, Palo Alto, 1995. Morgan Kaufmann.

[31] D.E. Goldberg. *Algorithmes génétiques. Exploration, optimisation et apprentissage automatique*. Addison-Wesley France, 1994.

[32] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-Organized Shortcuts in the Argentine Ant. *Naturwissenchaften*, 76 :579–581, 1989.

[33] G. Harik. Linkage learning in via probabilistic modeling in the EcGA. Technical Report 99010, IlliGAL, 1999.

[34] G. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. In *IEEE Conference on Evolutionary Computation*, pages 523–528, 1998.

[35] W. K. Hastings. Monte Carlo sampling method using Markov chains and their applications. *Biometrika*, 57, 1970.

[36] J. H. Holland. Outline for logical theory of adaptive systems. *J. Assoc. Comput. Mach.*, 3 :297–314, 1962.

[37] K. Hukushima and K Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *J. Phys. Soc. Jpn.*, 65 :1604–1608, 1996.

[38] Y. Iba. Population Annealing : An approach to finite-temperature calculation. In *Joint Workshop of Hayashibara Foundation and SMAPIP*. Hayashibara Forum, 2003.

[39] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598) :671–680, 1983.

[40] W. Krauth. *Advances in Computer Simulation*, chapter Introduction To Monte Carlo Algorithms. Springer-Verlag, 1998.

[41] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation*. Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, 2002.

[42] F. Liang and W. H. Wong. Evolutionary Monte Carlo : Application to C_p Model Sampling and Change Point Theorem. *Statistica Sinica*, 10, 2000.

[43] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21 :1087–1092, 1953.

[44] N. Monmarché, E. Ramat, G. Dromel, M. Slimane, and G. Venturini. On the similarities between AS, BSC and PBIL : toward the birth of a new meta-heuristics. E3i 215, Rapport interne 215, Université de Tours, 1999.

[45] N. Monmarché, N. Ramat, L. Desbarat, and G. Venturini. Probabilistic search with genetic algorithms and ant colonies. In A.S. Wu, editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop*, pages 209–211, 2000.

[46] H. Mühlenbein and G. Paaβ. From recombination of genes to the estimation of distributions I. Binary parameters. *Lecture Notes in Computer Science 1411 : Parallel Problem Solving from Nature*, PPSN IV :178–187, 1996.

[47] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7 :308–313, 1965.

[48] M. E. J. Newman and R. G. Palmer. Error estimation in the histogram Monte Carlo method. *arxiv :cond-mat/9804300*6, 1998.

[49] Y Okamoto and U. H. E. Hansmann. Thermodynamics of helix-coil transitions studied by multicanonical algorithms. *J. Phys. Chem.*, 99 :11276–11287, 1995.

[50] S. H. Pourtakdoust and H. Nobahari. An Extension of Ant Colony System to Continuous Problems. In M Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 294–301. Springer, September 2004.

[51] I. Rechenberg. *Cybernetic Solution Path of an Experimental Problem*. Royal Aircraft Establishment Library Translation, 1965.

[52] M.G.C. Resende. Greedy randomized adaptive search procedures (GRASP). Technical Report TR 98.41.1, AT&T Labs-Research, 2000.

[53] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *Parallel Problem Solving from Nature - PPSN V*, pages 418–427. Springer-Verlag, 1998.

[54] K. Socha. ACO for Continuous and Mixed-Variable Optimization. In M Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and

T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 25–36. Springer, September 2004.

[55] G. Syswerda. Simulated Crossover in Genetic Algorithms. In L. D. Whitley, editor, *Second workshop on Foundations of Genetic Algorithms*, pages 239–255, San Mateo, California, 1993. Morgan Kaufmann.

[56] É. D. Taillard. *Programmation à mémoire adaptative et algorithmes pseudo-gloutons : nouvelles perspectives pour les méta-heuristiques*. Thèse d'habilitation à diriger les recherches, Université de Versailles Saint Quentin en Yvelines, France, 1998.

[57] É. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive Memory Programming : A Unified View of Meta-Heuristics. *European Journal of Operational Research*, 135(1) :1–16, 1998.

[58] E.-G. Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, 8(5) :541–564, August 2002.

[59] E. Triki, Y. Collette, and P. Siarry. A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research*, 166 :77–92, 2005.

[60] O. Wendt and W. König. Cooperative Simulated Annealing : How Much Cooperation is Enough ? Technical Report 97-19, Institute of Information Systems, Goethe University, Frankfurt, 1997.