RAIRO Operations Research
RAIRO Oper. Res. **35** (2001) 329-365

ÉTAT DE L'ART DES MÉTHODES D'"OPTIMISATION GLOBALE"

Gérard Berthiau[1] et Patrick Siarry[2]

Communiqué par Gérard Plateau

Abstract. We present a review of the main "global optimization" methods. The paper comprises one introduction and two parts. In the introduction, we recall some generalities about non linear constraint-less optimization and we list some classifications which have been proposed for the global optimization methods. We then describe, in the first part, various "classical" global optimization methods, most of which available long before the appearance of Simulated Annealing (a key event in this field). There exists plenty of papers and books dealing with these methods, and studying in particular their convergence properties. The second part of the paper is devoted to more recent or atypical methods, mostly issued from combinatorial optimization. The three main methods are "metaheuristics": Simulated Annealing (and derived techniques), Tabu Search and Genetic Algorithms; we also describe three other less known methods. For these methods, theoretical studies of convergence are less abundant in the literature, and the use of convergence results is by far more limited in practice. However, the fitting of some of these techniques to continuous variables problems gave very promising results; that question is not discussed in detail in the paper, but useful references allowing to deepen the subject are given.

Mots clés : Global optimization, metaheuristics, convergence, continuous optimization.

Reçu en juillet 1999. Accepté en septembre 2001.

[1] Université de Nantes, CRTT-GE44, boulevard de l'Université, BP. 406, 44602 Saint-Nazaire, France ; e-mail : gerard.berthiau@ge44.univ-nantes.fr

[2] Université de Paris 12, LERISS, 61 avenue du Général de Gaulle, 94010 Créteil, France ; e-mail : siarry@univ-paris12.fr

1. Introduction

Durant les 20 dernières années, le domaine de la recherche sur l'"optimisation globale" s'est considérablement enrichi grâce, notamment, à l'accroissement de la puissance de calcul des ordinateurs. Ces progrès ont permis de résoudre des problèmes auparavant insolubles. Nous considérerons, dans ce chapitre, la date d'invention du recuit simulé [43] comme cruciale. En effet, avant cette date, les classes de méthodes employées pour la résolution des problèmes en variables continues, relevant de l'optimisation globale, étaient en grande partie distinctes de celles employées en optimisation combinatoire : le recuit simulé est la première méthode d'importance qui s'est largement répandue dans les deux familles de problèmes d'optimisation.

Dans une première partie, après quelques notions générales sur l'optimisation non linéaire sans contrainte, diverses méthodes d'optimisation globale, la plupart antérieures au recuit simulé et considérées comme "classiques", seront présentées. Il existe une abondante littérature qui traite de ces méthodes, étudie leur convergence et les tests d'arrêt associés.

Puis, nous focaliserons notre attention sur quelques méthodes plus récentes ou atypiques, issues, pour une grande part, de l'optimisation combinatoire : les trois principales d'entre elles sont des "métaheuristiques", la méthode du recuit simulé et ses variantes, la méthode tabou et les algorithmes génétiques ; nous évoquerons aussi trois autres méthodes de moindre notoriété. Pour ces méthodes, les références sont moins nombreuses ; la théorie de leur convergence, quand elle existe, est plus difficile à appliquer en pratique. Cependant, l'application de certaines de ces méthodes aux problèmes à variables continues a donné des résultats très encourageants, ce qui en fait un champ d'investigations prometteur.

L'adaptation des métaheuristiques à l'optimisation globale est un sujet de recherche en soi ; en effet, cette adaptation pose un problème spécifique pour chaque métaheuristique, et les démarches publiées sont variées. La présentation détaillée de ce sujet sort du cadre de cet article, qui vise une vue d'ensemble des méthodes d'optimisation globale, mais les principales références utiles pour approfondir le sujet sont citées.

1.1. Quelques notions générales sur l'optimisation globale [39]

Soit $\mathbf{x}$, un vecteur de dimension n dont les composantes x_i vérifient $a_i \leq x_i \leq b_i$, $i = 1, \ldots, n$, où a_i et b_i sont les composantes de 2 vecteurs $\mathbf{A}$ et $\mathbf{B}$, de dimension n, donnés. $\mathbf{A}$ et $\mathbf{B}$ définissent un domaine hyperrectangulaire que l'on notera S. Soit $f : \mathfrak{R}^n \to \mathfrak{R}$, une fonction à valeurs réelles. Le problème considéré est celui de trouver le ou un optimum global $\mathbf{x}^*$ de f c'est-à-dire $\mathbf{x}^*$ tel que :

$$\begin{array}{ll} & \forall \mathbf{x} \in S \subseteq \mathfrak{R}^n,\ f(\mathbf{x}) \geq f(\mathbf{x}^*) \quad \text{pour un minimum global} \\ \text{ou} & \forall \mathbf{x} \in S \subseteq \mathfrak{R}^n,\ f(\mathbf{x}) \leq f(\mathbf{x}^*) \quad \text{pour un maximum global.} \end{array} \tag{1}$$

Dans la suite de cette présentation, nous nous intéresserons seulement à la recherche des minimums d'une fonction, puisqu'un problème de maximisation peut

toujours se ramener à un problème de minimisation.

$$f(\mathbf{x}^*) = \mathbf{y}^* = \min_{x \in S}\{f(x)\} \cdot \tag{2}$$

Il est à noter que beaucoup de méthodes "classiques" pour l'optimisation globale s'appuient sur des conditions mathématiques précises afin de garantir leur succès, alors que les heuristiques donnent moins de certitudes quant aux résultats, mais sont moins restrictives : elles requièrent seulement la capacité d'évaluer la fonction à optimiser en un point quelconque de l'espace des solutions. C'est pourquoi nous présentons deux hypothèses classiquement rencontrées, l'une concernant le domaine de recherche de l'optimum global et exprimant les contraintes "de boîtes" sur les composantes de la solution, l'autre posant des conditions fortes sur la fonction objectif :

Hypothèse $\mathbf{(H_1)}$: l'hyperrectangle $S \subseteq \mathfrak{R}^n$ est convexe et compact.
Hypothèse $\mathbf{(H_2)}$: f est continue et possède des dérivées partielles premières $\partial f/\partial x_i$ et secondes $\partial^2 f/\partial x_i \partial x_j$ continues pour tout $\mathbf{x} \in \mathfrak{R}^n$.

L'hypothèse $\mathbf{(H_2)}$ est faite dans la plupart des méthodes "classiques" utilisant une procédure de raffinement local. Sous $\mathbf{(H_2)}$, des conditions nécessaires pour que $\mathbf{x}^*$ soit un minimum (local ou global) de f sont :

- $\nabla f(\mathbf{x}^*) = 0$ (stationnarité) ;
- le hessien $\nabla^2 f(\mathbf{x}^*) = [\partial^2 f/\partial x_i \partial x_j(\mathbf{x}^*)]$ est une matrice semi-définie positive (c'est-à-dire que $\forall \mathbf{y} \in \mathfrak{R}^n$, $\mathbf{y}^T \cdot \nabla^2 f(\mathbf{x}^*) \cdot \mathbf{y} \geq 0$).

Des conditions suffisantes pour que $\mathbf{x}^*$ soit un minimum (local ou global) de f sont :

- $\nabla f(\mathbf{x}^*) = 0$ (stationnarité) ;
- le hessien $\nabla^2 f(\mathbf{x}^*)$ est une matrice définie positive (*i.e.* $\forall \mathbf{y} \in \mathfrak{R}^n$, $\mathbf{y} \neq 0$, $\mathbf{y}^T \cdot \nabla^2 f(\mathbf{x}^*) \cdot \mathbf{y} > 0$).

Ces 2 conditions reviennent à supposer que f est strictement convexe dans un voisinage de $\mathbf{x}^*$.

Dans le cas d'une fonction convexe, la stationnarité à elle seule constitue une condition nécessaire et suffisante d'optimalité globale.

Puisqu'une procédure numérique ne peut pas fournir mieux qu'une réponse approchée, un élément $\mathbf{x}$ d'un des ensembles suivants sera considéré comme solution du problème [27], $\mathbf{x}^*$ étant un minimum global et ε un nombre positif quelconque :

$$A_x(\varepsilon) = \{\mathbf{x} \in S; \| \mathbf{x} - \mathbf{x}^* \| \leq \varepsilon\} \tag{3}$$

$$A_f(\varepsilon) = \{\mathbf{x} \in S; \| f(x) - f(\mathbf{x}^*) \| \leq \varepsilon\} \cdot \tag{4}$$

Dans le chapitre 1 de [39] ("Conditions for global optimality", de Hiriart–Urruty), sont rassemblés des *résultats théoriques récents*, relatifs à des problèmes d'optimisation globale dotés d'une structure particulière. Quelques uns de ces résultats ont

donné naissance à de nouveaux algorithmes d'optimisation globale. Ces résultats sont résumés dans les quatre paragraphes suivants :

- exploitation de l'enveloppe convexe de la fonction objectif ;
- cas des problèmes quadratiques ;
- minimisation "diff-convexe" ;
- intégration de la fonction objectif.

Exploitation de l'enveloppe convexe de la fonction objectif

Une condition nécessaire et suffisante (C.N.S.) pour que $\mathbf{x}^*$ soit un minimum global de f peut être obtenue en faisant appel à son "enveloppe convexe" cof ([39], Chap. 1). Sous des hypothèses peu restrictives, précisées dans [39], cof peut être définie comme la plus grande fonction convexe minorant f.

On admettra alors le théorème suivant :

Soit $f : \mathfrak{R}^n \to \mathfrak{R}$ une fonction différentiable sur $\mathfrak{R}^n$.
Alors $\mathbf{x}^*$ est un minimum global de f sur $\mathfrak{R}^n$ si et seulement si :
(i) $\nabla f(\mathbf{x}^*) = 0$ (stationnarité de $\mathbf{x}^*$)
(ii) $(\text{cof})(\mathbf{x}^*) = f(\mathbf{x}^*)$

La réalisation de la condition (ii) est délicate à tester en pratique, car la valeur exacte de $(\text{cof})(x^*)$ n'est pas simple à déterminer. Par contre, ce théorème peut être utilisé facilement sous sa forme négative : un point $\mathbf{x}^*$, stationnaire ou non, en lequel on trouve $l < f(\mathbf{x}^*)$, où l désigne un majorant de $(\text{cof})(\mathbf{x}^*)$, ne peut pas être un minimum global de f.

Il existe aussi une forme plus générale de ce théorème, applicable à une certaine classe de fonctions non différentiables partout ([39], Chap. 1).

Cas des problèmes quadratiques

Lorsque la fonction objectif et les contraintes d'un problème d'optimisation sont *quadratiques*, il est parfois possible d'obtenir un jeu de C.N.S. caractérisant les optimums globaux du problème.

Par exemple, on dispose du théorème suivant, établi dans le chapitre 1 de [39] :

Soit à minimiser $f(x) = (1/2).\ \langle Ax, x\rangle + \langle b, x\rangle$, sous la contrainte d'appartenance de x à une boule euclidienne : $x \in C = \{x \in \mathfrak{R}^n \,|\, \| x \| \leq \delta\}$
(A : matrice carrée symétrique d'ordre n, $b \in \mathfrak{R}^n$, $\delta > 0$)
Le point $\mathbf{x}^* \in C$ est un minimum global de f sur C si et seulement si $\exists \mu \geq 0$:
$(A + \mu.I_n)\mathbf{x}^* + b = 0$
$\mu(\| \mathbf{x}^* \| - \delta) = 0$
$A + \mu.I_n$ est semi-définie positive
où I_n désigne la matrice identité d'ordre n.

Ce résultat est intéressant, car de nombreux algorithmes (de type Newton par exemple) opèrent sur des approximations quadratiques locales de la fonction objectif.

Minimisation "diff-convexe"

La convexité de la fonction objectif est une caractéristique importante en optimisation, car elle permet d'étendre à tout l'espace la validité d'une propriété locale : un minimum local peut ainsi s'avérer global. Lorsque la fonction objectif f est non convexe, elle peut parfois être exprimée comme la différence entre deux fonctions convexes, soit g et h : dans ce cas, la minimisation de $f = g - h$, sur un ensemble convexe fermé, admet des C.N.S. d'optimalité globale.

Dans ce domaine, dénommé minimisation "diff-convexe" (ou d.c.), on peut ainsi établir notamment (Chap. 1 de [39]) le théorème suivant :

Soit $f = g - h$, où g et h sont convexes (pour la fonction g, une propriété plus faible peut même être substituée à la convexité [39]).
Alors $\mathbf{x}^*$ est un minimum global de f sur $\mathfrak{R}^n$ si et seulement si $\partial_\varepsilon h(\mathbf{x}^*) \subset \partial_\varepsilon g(\mathbf{x}^*)$, $\forall \varepsilon > 0$.
Cette condition utilise la notion d'ε sous-différentielle d'une fonction φ en $\mathbf{x}^*$, soit $\partial_\varepsilon \varphi(\mathbf{x}^*)$, définie comme l'ensemble des $s \in \mathfrak{R}^n$ vérifiant : $\varphi(\mathbf{x}) \geq \varphi(\mathbf{x}^*) + \langle s, \mathbf{x} - \mathbf{x}^* \rangle - \varepsilon$, $\forall \mathbf{x} \in \mathfrak{R}^n$.

Intégration de la fonction objectif

Les résultats précédents concernaient des problèmes d'optimisation particuliers. Dans le cas général de fonctions objectifs continues quelconques, il n'existe pas de critère d'optimalité réellement exploitable. Cependant, *l'intégration* de la fonction objectif peut rendre, dans un certain sens, le problème convexe, et de là procurer des résultats intéressants, quoique de portée purement théorique (voir dans [39], Chap. 1).

Remarque. Dans les "problèmes d'optimisation difficiles" en variables continues rencontrés en pratique – par exemple lorsqu'on cherche à optimiser les performances des circuits électroniques [7] – la propriété de convexité de la fonction objectif est généralement absente, ce qui entraîne l'existence d'un grand nombre d'optimums locaux. Cette situation, semblable à celle rencontrée dans les problèmes combinatoires à grand nombre de variables, suggère l'adaptation des métaheuristiques d'origine combinatoire, capables d'éviter le blocage dans un optimum local.

1.2. Classifications des méthodes d'"optimisation globale"

Dans la littérature, les méthodes développées pour résoudre des problèmes d'optimisation globale sont réparties en différentes classes selon les auteurs [74] :

- Dixon *et al.* proposent 2 classes selon que les méthodes utilisent ou non des éléments stochastiques [26, 27].
  - les *méthodes déterministes*, qui n'utilisent aucun concept stochastique, impliquent des hypothèses supplémentaires sur la fonction f à optimiser,

telles que f soit continûment dérivable, ou que f soit lipschitzienne, *i.e.* :

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in S, \mid f(\mathbf{x}_1) - f(\mathbf{x}_2) \mid \leq L \parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel \quad (\text{où } L = \text{constante}). \tag{5}$$

Malheureusement, en pratique, il s'avère difficile de savoir si la fonction objectif f satisfait ou non à de telles conditions, les critères d'arrêt des algorithmes liés à ces conditions conduisant à des temps de calcul importants.

- les *méthodes stochastiques* pour lesquelles la procédure de minimisation dépend en partie d'événements probabilistes. L'inconvénient majeur de ces méthodes est qu'on ne peut garantir leur convergence que d'une manière asymptotique (ou pas du tout...). Cette distinction déterministe/stochastique n'est pas toujours très nette et la plupart des algorithmes d'optimisation globale proposés combinent ces deux approches.
- Rinnooy Kan *et al.* construisent 5 classes fondées sur la philosophie sous-jacente des méthodes [59] :
  - *partitionnement et recherche* : S est découpé en sous-régions plus petites parmi lesquelles le minimum global est recherché, dans l'esprit des méthodes par séparation et évaluation de l'optimisation combinatoire ;
  - *approximation et recherche* : f est remplacée par des approximations qui s'affinent au fur et à mesure du processus d'optimisation ;
  - *énumération des minimums locaux* : si celle-ci pouvait être exhaustive, cette méthode résoudrait le problème de minimisation globale posé ;
  - *amélioration des minimums locaux* : en exploitant les capacités d'un processus de recherche locale efficace, une suite de minimums locaux de valeurs décroissantes est générée. Le dernier terme de la suite est le minimum "global" ;
  - *décroissance globale de la fonction de coût* : amélioration permanente de la fonction f ; conduit, à la fin du processus, au minimum "global".
- Törn *et al.* s'appuient sur la précision des méthodes pour les diviser en deux classes principales [74] :
  - les *méthodes garantissant l'exactitude* du minimum global, ou méthodes de couverture ("covering methods"). Ces méthodes sont basées sur la recherche puis l'élimination des sous-régions ne contenant pas le minimum global. La *méthode des intervalles* (*cf.* Sect. 2.2) et la méthode par séparation et évaluation font partie de cette classe.
  - les *méthodes sans garantie sur la précision*, elles-mêmes divisées comme suit :
  - les *méthodes directes*, n'utilisant qu'une information locale (évaluation de la fonction f).
    Cette classe comprend trois subdivisions :
    - ♦ les méthodes de recherche aléatoire (*recherche aléatoire pure*, *méthode "multistart"* (*cf.* Sect. 2.1)) ;
    - ♦ les *méthodes de regroupement* ("clustering methods" (*cf.* Sect. 2.3)) ;

  - ♦ les méthodes de descente généralisée comme les *méthodes de trajectoires* (*cf.* Sect. 2.4) et les méthodes de pénalisation (*e.g.* *méthode de "percement de tunnel"* (*cf.* Sect. 2.5)) ;
- – les *méthodes indirectes* ou méthodes d'échantillonnage, pour lesquelles l'information locale (échantillon de points) est utilisée pour construire un modèle statistique global de la fonction objectif, méthodes divisées en deux sous-classes :
  - ♦ les méthodes approchant les *ensembles de niveaux* ("level sets" (*cf.* Sect. 2.6)) ;
  - ♦ les méthodes approchant la fonction objectif.

Sans chercher à choisir une classification plutôt qu'une autre, nous allons, dans une première partie, examiner quelques-unes des méthodes classiques [56], puis, en second lieu, nous nous intéresserons à un groupe de méthodes apparues durant les 20 dernières années pour résoudre "au mieux" les problèmes dits "d'optimisation difficile". Dans la littérature, deux sortes de problèmes reçoivent cette appellation, non définie strictement (et liée, en fait, à l'état de l'art en matière d'optimisation) :

- certains problèmes d'optimisation combinatoire, pour lesquels on ne connaît pas d'algorithme exact "rapide" (c'est le cas, en particulier, des problèmes NP-difficiles [3]) ;
- certains problèmes d'optimisation à variables continues, pour lesquels on ne connaît pas d'algorithme permettant de repérer un optimum global à coup sûr et en un nombre fini de calculs.

Un grand nombre d'"heuristiques", qui produisent des solutions suboptimales, ont été développées pour les problèmes d'optimisation combinatoire difficile. La plupart d'entre elles sont adaptées à la résolution d'un type de problème donné. D'autres, au contraire, parfois appelées "méta-heuristiques" [15], sont capables de s'adapter à différents types de problèmes, combinatoires ou même continus. Ces "méta-heuristiques", ont en commun, en outre, la plupart des caractéristiques suivantes :

- elles sont pour la plupart stochastiques (c'est une approche naturelle pour faire face à "l'explosion combinatoire" des possibilités) ;
- en raison de leur origine combinatoire, elles ne peuvent s'appliquer aux problèmes continus qu'après transformation (plus ou moins aisée ...) ;
- du fait de cette origine, elles ont l'avantage d'être directes, c'est-à-dire sans gradient ;
- certaines sont inspirées par des analogies : avec la biologie (algorithmes génétiques, réseaux de neurones) ou avec la physique (recuit simulé).

Ces méthodes ne s'excluent pas mutuellement : en effet, dans l'état actuel de la recherche, il est le plus souvent impossible de prévoir avec certitude l'efficacité d'une méthode donnée, quand elle est appliquée à un problème donné [33].

Pour une bonne part d'entre eux, les algorithmes stochastiques sont itératifs et sont constitués de trois étapes principales qui sont appliquées à partir d'un point initial de la fonction de coût choisi aléatoirement, sans connaissance particulière : une perturbation aléatoire, un critère d'acceptation et un critère d'arrêt. Dans

chacune de ces étapes, des choix différents peuvent être effectués. Sans prétendre être exhaustive, la liste suivante présente quelques possibilités pour ces étapes :

- la perturbation aléatoire :
  - toutes les coordonnées du vecteur solution courant, ou une partie seulement, sont perturbées, toutes à la fois ou itérativement ;
  - la perturbation suit une loi de distribution particulière dans le voisinage du point courant avec une matrice de variance/covariance déterminée, ou suit une distribution uniforme ;
  - le ou les nouveaux points sont générés selon une loi qui dépend des points précédents et/ou des valeurs de la fonction, ou ne dépend que du point courant;
  - etc.
- le critère d'acceptation :
  - le ou les nouveaux points sont acceptés selon une certaine loi de probabilité ;
  - le ou les meilleurs des nouveaux points sont conservés pour l'étape suivante ;
  - une recherche locale est effectuée à partir du ou des nouveaux points ;
  - etc.
- les critères d'arrêt :
  - ils peuvent être liés à la qualité du minimum obtenu :
    - la procédure est arrêtée quand il n'y a plus d'améliorations de la solution après un certain nombre d'itérations ;
    - elle s'arrête quand les perturbations ne dépassent plus un certain seuil ;
  - ils peuvent être liés à des coûts de calcul :
    - la procédure est stoppée après un nombre fixé d'évaluations de la fonction objectif ou d'itérations, ou après un temps de calcul fixé ;
    - etc.

2. Méthodes "classiques"

2.1. Les méthodes de recherche aléatoire

Trois méthodes simples sont représentatives de ce groupe : la méthode de recherche aléatoire pure, la méthode "singlestart" et la méthode "multistart". Pour chacune de ces méthodes, un N-échantillon est généré par une distribution uniforme sur la région S.

Pour la *méthode de recherche aléatoire pure* [2, 10], la fonction objectif est évaluée en chaque point de l'échantillon, et la solution $\mathbf{x}^*$ correspond à la plus petite valeur de f obtenue,

$$f^* = \min_i [f(\mathbf{x}_i)]. \tag{6}$$

Cette méthode est la plus simple des méthodes stochastiques. Elle possède toutefois un inconvénient rédhibitoire, lorsque le nombre de variables augmente : en général, la taille de l'échantillon nécessaire pour obtenir une solution proche de l'optimum global croît exponentiellement avec la dimension du problème [39].

Du fait de sa simplicité d'implémentation, la méthode de recherche aléatoire pure est souvent employée par les ingénieurs confrontés à des problèmes d'optimisation pratiques. Elle est également très utile pour procurer une borne de performance minimale, dans les études empiriques comparatives.

Dans la *méthode de direction aléatoire* [24], le $k^{\text{ième}}$ point $\mathbf{x}_k$ en lequel la fonction objectif est évaluée, est déterminé comme une fonction f d'un point ξ_k, obtenu à partir d'une distribution G_k et d'une autre fonction D du point de la précédente itération $\mathbf{x}_{k-1}$. La fonction f est de la forme suivante :

$$f(\mathbf{x}_k) = f(D(\mathbf{x}_{k-1}, \xi_k)) \leq \min\left(f(\mathbf{x}_{k-1}), f(\xi_k)\right). \tag{7}$$

La fonction D peut être prise de différentes façons : elle peut correspondre à la minimisation de $f(\alpha\mathbf{x}_{k-1} + (1-\alpha)\xi_k)$, $\alpha \in [0,1]$, ou à la minimisation d'une fonction polynomiale quelconque approchant f, ou, plus simplement encore, elle peut prendre la valeur de $\min\left[f(\mathbf{x}_{k-1}), f(\xi_k)\right]$.

Notons que la méthode de direction aléatoire est une généralisation de la méthode de recherche aléatoire pure lorque l'on prend pour G_k, $\forall k$, la distribution uniforme sur S et :

$$D(\mathbf{x}_{k-1}, \xi_k) = \min(f(\mathbf{x}_{k-1}), f(\xi_k)).$$

On peut prouver que, *si* la distribution G_k satisfait :

$$\Pi_{k=1}^{\infty}(1 - G_k(A)) = 0 \tag{8}$$

pour chaque $A \subseteq S$ avec la mesure de Lebesgue[3], $m(A) > 0$, *alors*, un point arbitrairement proche, *i.e.* à $\varepsilon > 0$ près, du minimum global sera trouvé avec une probabilité qui tend vers 1 quand $k \to \infty$.

Les deux méthodes qui suivent sont des variantes d'une technique dans laquelle, à chaque itération, les points sont échantillonnés à partir d'une distribution uniforme sur S durant une phase globale ; puis une phase locale de recherche de minimum est effectuée. Cela nécessite qu'une quelconque procédure locale de descente, partant d'un point initial arbitraire $\mathbf{x}$ de la région de recherche S, soit capable de fournir un minimum local $\hat{\mathbf{x}}^*$ (de nombreuses procédures de cette sorte existent dans la littérature sur la "programmation mathématique" [51] : elles supposent généralement que le nombre de minimums locaux est fini, et que f est deux fois continûment dérivable).

[3]La mesure de Lebesgue définie sur $\Re^n$ est la mesure unique telle que tout pavé borné S a pour mesure $m(S) = \Pi_{i=1}^{n}(b_i - a_i)$, dès que S est le produit cartésien des intervalles bornés de $\Re$ de bornes a_i et b_i (cette mesure est la plus intuitive, elle correspond à la longueur du domaine S pour $n = 1$, sa surface pour $n = 2$, son volume pour $n = 3$, ...).

Pour la *méthode "singlestart"*, la recherche locale est effectuée à partir du meilleur point de l'échantillon. Soit PML, une Procédure de Minimisation Locale,

$$\min[f(\mathbf{x}_i)] \xrightarrow{\text{PML}} \hat{f}^*, \text{ où } \hat{f}^* \text{ est, au moins, un minimum local.} \tag{9}$$

La *méthode "multistart"* effectue une PML à partir de chaque point de l'échantillon et f^* est le meilleur minimum obtenu.

$$\begin{aligned} &\forall i, \mathbf{x}_i \xrightarrow{\text{PML}} \hat{f}_k^* \\ &f^* = \min\left[\hat{f}_k^*\right]. \end{aligned} \tag{10}$$

La convergence de ces algorithmes très simples est asymptotique, c'est-à-dire que la probabilité d'atteindre le minimum global de f tend vers 1 lorsque le nombre de points échantillonnés tend vers l'infini. Il est à noter que ces méthodes cherchent indifféremment dans des régions "prometteuses" ou non et qu'un grand nombre de recherches locales de la méthode "multistart" aboutissent à des minimums locaux déjà obtenus. D'où un effort de calcul en partie inutile.

2.2. Les méthodes de couverture

Les méthodes de couverture ("covering methods") reposent sur une hypothèse, qui est satisfaite pour la plupart des problèmes pratiques : la fonction objectif est à taux de variation borné (hypothèse plus restrictive que la dérivabilité). Dans ce cas, l'évaluation de f aux seuls nœuds d'un maillage suffisamment serré du domaine S garantit, en principe, la détection, avec une précision donnée, du minimum global. L'inconvénient de la démarche réside dans le nombre élevé de noeuds requis, même pour un problème de dimension modeste.

Les méthodes les plus simples sont fondées sur la détection et l'élimination de sous-régions ne contenant pas le minimum global. Un point de vue complémentaire consiste en une procédure de localisation et de découpage qui construit des petits intervalles (avec la précision voulue) autour des minimums. Une autre procédure fournit des approximations locales de plus en plus précises de la fonction objectif.

La méthode des intervalles [52, 53] utilise le principe de séparation/évaluation ("branch and bound"). L'algorithme effectue, à chaque itération, une division du domaine S en sous-domaines qui peuvent être de tailles différentes. Le principe de séparation est appliqué : à chaque itération, la recherche est effectuée dans le sous-domaine où la fonction objectif a la valeur la plus basse, en partant du principe que les chances de trouver le minimum global sont meilleures dans ce sous-domaine.

L'approche des intervalles a fait l'objet d'importants progrès au cours de la dernière décade. Elle s'applique désormais à l'optimisation avec ou sans contraintes, sur des domaines bornés ou non.

2.3. Les méthodes de regroupement [60]

Pour ces méthodes, un algorithme d'analyse des groupes est utilisé pour éviter de retrouver plusieurs fois un minimum déjà connu (comme dans les méthodes de recherche aléatoire). L'idée est de déterminer la région d'attraction d'un minimum. Ainsi, à partir d'un point appartenant à cette région, on évitera de relancer la PML, qui aboutirait au même minimum. Démarrant d'un échantillonnage uniforme de S, des groupes de points mutuellement proches appartenant à une même région d'attraction sont créés et la PML est lancée une seule fois dans chaque région. Deux stratégies permettent de bâtir ces regroupements :

- la *réduction* est basée sur l'idée de ne retenir que les points ayant une valeur de f assez basse (ceci grâce à un seuil ε arbitrairement choisi) ; ces points formeront des groupes autour des minimums [5] ;
- la *concentration* consiste à rapprocher chacun des points précédents du minimum le plus proche à l'aide de quelques itérations d'une PML [73].

Chaque itération est constituée d'une de ces deux stratégies lors de laquelle les points sont répartis dans les différents groupes, suivie d'une série de PML : pour chaque point fournissant une faible valeur de f et n'ayant été placé dans aucun groupe existant, une PML est effectuée. Malheureusement, ces transformations ne fournissent pas nécessairement des groupes de points qui correspondent à des régions d'attraction des minimums de f.

2.4. Une méthode de descente généralisée : la méthode de la trajectoire [9]

Le développement des méthodes de minimisation locale a fait l'objet de très nombreux travaux. Il est donc naturel d'essayer de généraliser ou de modifier ces méthodes en vue de l'optimisation globale. Deux approches ont été proposées pour cette "descente généralisée". La première, exposée dans ce paragraphe, est la méthode de la trajectoire : elle repose sur une modification de l'équation différentielle décrivant la trajectoire de descente locale. La seconde consiste à effectuer, en alternance, des descentes locales classiques et des modifications appropriées de la fonction objectif : elle sera évoquée dans le paragraphe suivant.

La méthode de la trajectoire est fondée sur la construction par intégration numérique des chemins le long desquels le gradient de la fonction objectif pointe dans une direction constante. En effet, une propriété importante des équations différentielles est que la trajectoire, c'est-à-dire le lieu de la solution en fonction du temps, passe par le voisinage de la plupart des points stationnaires de la fonction objectif. Déterminer tous les minimums locaux de f, et donc le minimum global, revient à résoudre le système d'équations $\nabla f(\mathbf{x}) = 0$ où ∇f est le gradient de f. On reprend l'hypothèse **(H2)** sur la différentiabilité de f. La solution de ce système peut être obtenue en utilisant les trajectoires du système d'équations

différentielles suivant :

$$\frac{\mathrm{d}\nabla f\left(\mathbf{x}(t)\right)}{\mathrm{d}t} \pm \nabla f\left(\mathbf{x}(t)\right) = 0. \tag{11}$$

Avec $\dot{\mathbf{x}} = \dfrac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$, ce système peut être écrit sous la forme :

$$\frac{\partial \nabla f(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}} \pm \nabla f(\mathbf{x}) = 0, \tag{12}$$

donnant :

$$\dot{\mathbf{x}} = \pm \left(\frac{\partial \nabla f(\mathbf{x})}{\partial \mathbf{x}} \right)^{-1} \nabla f(\mathbf{x}), \tag{13}$$

où $\dfrac{\partial \nabla f(\mathbf{x})}{\partial \mathbf{x}}$ est le Jacobien du gradient de f, soit le Hessien de f.

En utilisant (11), on obtient le gradient $\nabla f(\mathbf{x}(t)) = \nabla f(\mathbf{x}(0)) \exp(\pm t)$, ce qui signifie que le gradient possède une direction constante sur toute la trajectoire. Ceci est très utile pour la correction du pas d'intégration numérique de (13). Le signe de (13) change quand la trajectoire traverse le voisinage des points de dégénérescence de la matrice $\dfrac{\partial \nabla f}{\partial \mathbf{x}}$ (c'est-à-dire des points en lesquels cette matrice possède des composantes nulles). La trajectoire est forcée artificiellement à travers ces points.

Malheureusement, cette méthode est connue pour être mise en défaut par certaines fonctions et les conditions de convergence vers le minimum global ne sont pas très claires. En outre, il est délicat de déceler les problèmes d'optimisation pour lesquels son emploi est indiqué. D'une manière générale, la stabilité de la méthode n'est pas assurée vis-à-vis des erreurs d'arrondi : son application est donc hasardeuse dans le cas où les expressions analytiques des dérivées de f ne sont pas disponibles.

2.5. Une méthode de pénalisation : la méthode de "percement de tunnel" [45–47]

L'algorithme "multistart" serait efficace si les déterminations multiples d'un même minimum local pouvaient être évitées. Dans les méthodes de pénalisation, ce phénomène est prohibé au moyen de fonctions de pénalité, affectées à chaque minimum local rencontré. Nous nous limitons ici à la présentation de l'une des méthodes de pénalisation : la méthode de "percement de tunnel".

La méthode de percement du tunnel ("tunneling"), présentée par Levy *et al.*, est composée d'une suite de cycles, chacun consistant en 2 phases :

- phase de minimisation ;
- phase de "percement du tunnel".

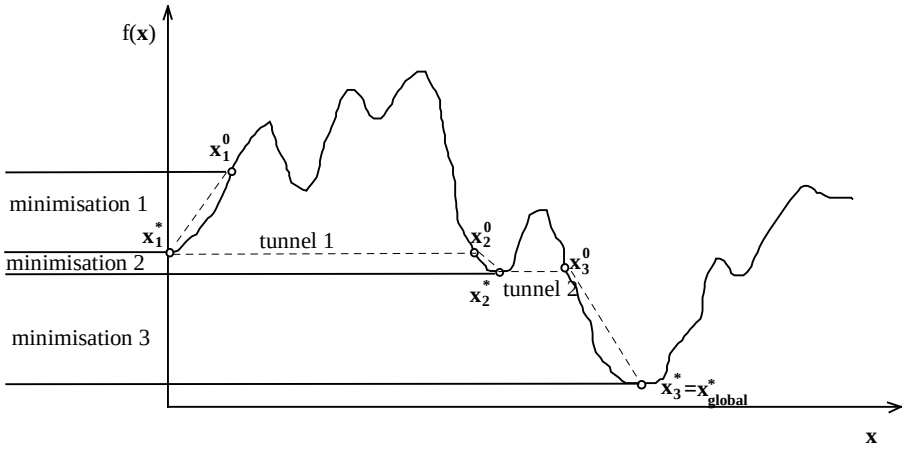

FIGURE 1. Interprétation géométrique de la méthode de percement du tunnel.

La phase de minimisation est exécutée afin d'obtenir un point $\mathbf{x}^*$ tel que les conditions suivantes soient satisfaites :

$$f'(\mathbf{x}^*) = 0 \text{ et } \mathbf{y}^T f''(\mathbf{x}^*)\mathbf{y} > 0. \tag{14}$$

Pour un point de départ donné $\mathbf{x}_0$, on peut utiliser n'importe quel algorithme de minimisation locale.

La phase de percement du tunnel est connue pour atteindre un "bon" point de départ pour la phase de minimisation suivante : partant du point $\mathbf{x}^*$ que l'on vient d'obtenir, on cherche une solution de l'équation de "percement" donnée par :

$$f(\mathbf{x}) - f(\mathbf{x}^*) = 0. \tag{15}$$

Dès que l'on atteint un zéro de (15) pour un $\mathbf{x}_0 \neq \mathbf{x}^*$, ce point est pris comme nouveau point de départ pour la phase de minimisation suivante. Si, après un temps de calcul suffisant (arbitrairement choisi), un $\mathbf{x}_0 \neq \mathbf{x}^*$ solution de (15) n'est pas trouvé, on considère que l'équation :

$$f(\mathbf{x}) - f(\mathbf{x}^*) > 0, \ \forall \mathbf{x} \in S, \ \mathbf{x} \neq \mathbf{x}^* \tag{16}$$

est satisfaite et l'algorithme est terminé. La figure 1 donne une représentation géométrique de cet algorithme pour une fonction unidimensionnelle.

Partant du point $\mathbf{x}_1^0$, la PML aboutit au minimum local $\mathbf{x}_1^*$, puis la phase de percement de tunnel fournit le point $\mathbf{x}_2^0$, qui sert de point de départ pour une seconde itération en deux phases, minimisation jusqu'à $\mathbf{x}_2^*$ puis tunnel jusqu'à $\mathbf{x}_3^0$. La dernière itération amène, dans la phase de minimisation, en $\mathbf{x}_3^*$, puis la phase de percement de tunnel ne pouvant fournir de zéro de l'équation (15), le programme s'arrête, considérant $\mathbf{x}_3^*$ comme le minimum global.

Cette description du principe de la méthode élude le mécanisme de pénalisation : en réalité, l'équation de percement comporte un terme de pénalisation adaptatif, qui est actualisé à chaque fois qu'un nouveau minimum local est détecté. Cette procédure permet d'éviter l'attraction des minimums locaux déjà rencontrés. Elle comporte toutefois un inconvénient : la fonction auxiliaire de percement, construite progressivement à partir de f et des coefficients de pénalité, devient de plus en plus plate, ce qui complique sa minimisation.

2.6. Les méthodes multi-niveaux [60]

L'objectif de ces méthodes est la détermination exhaustive des minimums locaux de f. Par conséquent, elles sont inefficaces lorsque le nombre de ces minimums est très élevé.

Ces méthodes combinent l'efficacité numérique des méthodes de regroupement et les avantages de la méthode "multistart". Elles sont toujours itératives, associant, comme dans les méthodes de regroupement, une phase globale d'échantillonnage à l'aide d'une distribution uniforme sur S, à une phase locale de descente, ce qui suppose que l'hypothèse **(H2)** soit vérifiée.

À chaque itération de la méthode "multilevel-single linkage" [58], pendant la phase d'échantillonnage, la fonction f est évaluée en chaque point de l'échantillon, puis durant la phase de minimisation, la PML n'est effectuée qu'à partir d'un sous-ensemble de l'échantillon. La sélection des points de départ de la PML est un aspect important de cette méthode : à chaque itération k, chaque point échantillonné $\mathbf{x}$ est sélectionné comme point de départ de la PML, s'il ne l'a pas déjà été lors d'une précédente itération, et s'il n'y a pas d'autre point $\mathbf{y}$ de l'échantillon qui vérifie :

$$\| \mathbf{x} - \mathbf{y} \| \leq r(k) \text{ et } f(\mathbf{x}) > f(\mathbf{y}) \cdot \tag{17}$$

La "distance critique" $r(k)$ est donnée par :

$$r(k) = \left(1/\sqrt{\pi}\right) \left[\Gamma\left(1 + \frac{n}{2}\right) m(S) \sigma \frac{\log kN}{kN}\right]^{1/n} \tag{18}$$

en désignant par : Γ la fonction gamma, σ une constante positive, N la taille de l'échantillon par itération, $m(S)$ la mesure de Lebesgue du domaine n-dimensionnel S.

Cette procédure de sélection peut être appliquée seulement aux γkN points de l'échantillon ayant les plus petites valeurs de f, $\gamma \in [0, 1]$, ce qui correspond à une réduction de la taille de l'échantillon. Cette réduction n'affecte pas la fiabilité théorique ni l'efficacité de la méthode "multi-level single linkage", mais apparaît plutôt comme une amélioration. Un test d'arrêt probabiliste détermine, enfin, si l'on procède ou non à une nouvelle itération.

Une autre possibilité est la méthode "multi-level mode analysis" : à l'étape k, l'échantillon de dimension N tiré sur S est partitionné en sous-ensembles,

ou cellules, de mesure égale à $(m(S)\sigma \log kN/kN)$, $\sigma > 0$. Après réduction de l'échantillon, les cellules qui contiennent plus de $(1/2\sigma \log kN)$ points sont considérées comme "pleines". À chaque cellule pleine, la plus petite valeur de f, pour tous les points de la cellule, est considérée comme valeur de la cellule elle-même. Enfin, la PML est appliquée seulement en un point de chaque cellule pleine, sauf si la cellule considérée possède, dans son voisinage, une cellule pleine ayant une valeur de f plus petite.

Tous ces algorithmes d'optimisation globale utilisent un critère d'arrêt le plus souvent probabiliste, qui réalise un compromis entre l'effort numérique nécessaire pour assurer la convergence théorique et l'obtention d'un coût de calcul raisonnable.

2.7. Autres méthodes "classiques"

Nous évoquons succinctement dans ce paragraphe les méthodes "classiques" de portée plus théorique, non traitées plus haut. Ces méthodes sont applicables seulement lorsque la fonction objectif possède une propriété structurelle particulière. Une présentation détaillée de ces méthodes se trouve dans [39].

La "programmation Diff-Convexe" ("DC programming") s'intéresse au cas où f peut s'exprimer comme la différence de deux fonctions convexes.

La "programmation quadratique" est, après la programmation linéaire, un des domaines les plus étudiés, car le caractère quadratique de f induit de nombreux résultats théoriques.

La "théorie Minimax", introduite par Von Neumann, a joué un rôle fondamental en optimisation convexe, et dans la théorie des jeux. Elle est également féconde en optimisation non convexe.

La "programmation multiplicative" s'intéresse à une classe de problèmes (de nature généralement économique) où intervient le produit d'un nombre fini de fonctions convexes, soit dans f, soit dans la définition du domaine réalisable.

L'"optimisation de Lipschitz" s'intéresse typiquement au cas où la fonction objectif n'est pas connue analytiquement, mais peut être évaluée au moyen de mesures : l'exemple le plus courant est celui de la caractérisation de modèles non linéaires.

La "programmation fractionnelle" (dite aussi "programmation hyperbolique") aborde le cas où f met en jeu un ou plusieurs rapports de fonctions.

L'"analyse d'intervalles", qui exploite classiquement l'arithmétique des intervalles pour évaluer les erreurs d'arrondi ou de troncature, procure aussi un cadre commode dans le contexte de l'optimisation, pour morceler le domaine de recherche. Les algorithmes d'optimisation globale fondés sur cette approche effectuent généralement des bissections successives du domaine de recherche, en procédant par "séparation et évaluation". Il existe une littérature très abondante sur le sujet : on pourra consulter en particulier les références [41] et [42].

3. Heuristiques récentes [57]

La plupart des méthodes présentées dans cette partie ont été utilisées, avec succès, pour des problèmes à variables discrètes, à haute combinatoire. Une difficulté supplémentaire s'ajoute quant à leur adaptation aux problèmes à variables continues.

Les trois premiers paragraphes sont consacrés aux métaheuristiques dominantes : la méthode du recuit simulé et ses variantes, la méthode tabou et les algorithmes génétiques. Nous ne traitons pas ici d'autres méthodes, comme les réseaux de neurones et la méthode GRASP ("Greedy Random Adaptive Search Procedure"), qui sont aussi considérées comme des métaheuristiques en optimisation combinatoire. En effet, l'emploi des réseaux de neurones à des fins d'optimisation est plus marginal et surtout, à notre connaissance, ces métaheuristiques n'ont pas été adaptées jusqu'ici à l'optimisation globale.

Nous avons enfin regroupé, dans le quatrième paragraphe, les présentations succinctes de quelques méthodes de moindre notoriété : la méthode de recherche distribuée, la méthode du bruitage et la méthode Aliénor.

3.1. La méthode du recuit simulé et ses variantes

3.1.1. *La méthode du recuit simulé [14, 43, 44]*

Nous présentons très succinctement, dans cette partie, la méthode du recuit simulé en en donnant simplement le principe. Une présentation détaillée de la théorie et des applications est donnée dans [67–69]. Cette méthode est issue d'une analogie entre le phénomène physique de refroidissement lent d'un solide en fusion, qui le conduit à un état cristallin de basse énergie, et la recherche de minimums globaux dans un problème d'optimisation. Elle exploite généralement l'algorithme de Metropolis [49].

L'algorithme de Metropolis est le critère d'acceptation d'une configuration (ou solution) $\mathbf{x}'$ du système construite en perturbant la configuration courante $\mathbf{x}$ ($\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}$). Il est présenté en figure 2 pour une "température" T donnée et en posant $f = f(\mathbf{x})$ et $f' = f(\mathbf{x}')$.

Le paramètre de contrôle T est la "température" du système. À très haute température, tous les changements sont acceptés : l'algorithme équivaut alors à une marche aléatoire dans l'espace des configurations. À T finie, lorsque l'équilibre thermodynamique est atteint [1,23], la probabilité d'accepter une configuration $\mathbf{x}_k$ est donnée par :

$$p(\mathbf{x}_k) = \frac{1}{Z} \exp\left(\frac{-f(\mathbf{x}_k)}{T}\right) \text{ avec } Z = \sum_k \exp\left(\frac{-f(\mathbf{x}_k)}{T}\right) \cdot$$

Cette distribution de Boltzman est modulée par la distribution des états, si celle-ci n'est pas uniforme. Lorsque la température est abaissée suffisamment lentement [48] pour que l'équilibre thermodynamique soit maintenu, le processus se

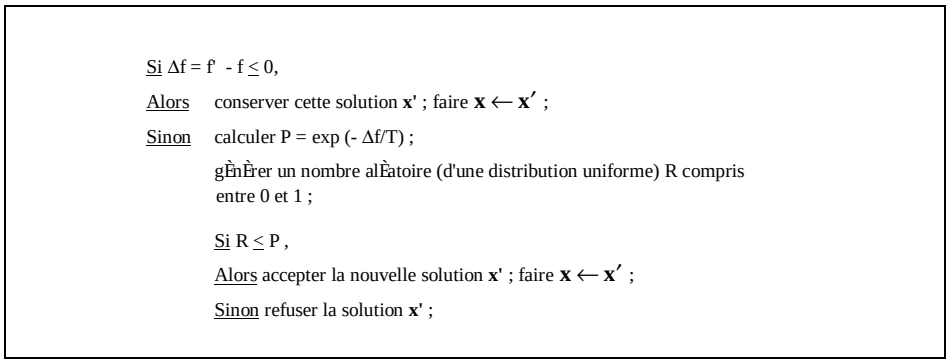

FIGURE 2. Règle de Metropolis.

traduit par une augmentation du poids des configurations de basse énergie. Nous présentons figure 3 l'algorithme du recuit simulé.

L'originalité de cet algorithme se situe dans l'étape 5). En effet, dans les méthodes classiques, on ne peut accepter aucune perturbation qui provoque une dégradation du système. Ici, une telle perturbation peut être acceptée avec une probabilité $p = \exp\left(\frac{-(f(\mathbf{x}')-f(\mathbf{x})}{T}\right)$. On accepte de perdre un peu pour gagner plus ultérieurement.

La présentation rapide que nous venons de faire du recuit simulé permet d'introduire plus loin ses variantes.

Différentes versions du recuit simulé ont été proposées dans la littérature pour l'optimisation de problèmes à variables continues. Elles diffèrent par la stratégie de discrétisation, définie par les éléments suivants :

- la fréquence de changement du pas ;
- la fréquence de variation du pas ;
- la loi de calcul du mouvement d'une variable.

Dans [7], les 5 principales méthodes publiées sont inventoriées et présentées :

- la méthode de Cerny [13, 14] ;
- la méthode de Corana *et al.* [19] ;
- la méthode de Vanderbilt *et al.* [75] ;
- la méthode du "recuit simulé généralisé" de Bohachevski *et al.* [8, 11] ;
- la méthode de Catthoor *et al.* [12].

La méthode élaborée par les auteurs de cet article est présentée en détail dans [70]. Elle a été mise au point empiriquement au moyen d'une batterie de fonctions analytiques [65, 66], dont les minimums locaux et globaux sont connus. Cette méthode a été appliquée notamment en électronique, d'une part pour la conception de circuits, d'autre part pour la caractérisation de modèles de composants [7, 54].

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser et évaluer la valeur de la fonction objectif f = f(**x**) ;
2) Choisir une température initiale "élevée" T.
3) Perturber cette solution pour obtenir une nouvelle solution **x' = x + Δx** ;
4) Calculer Δf =f(**x'**) - f(**x**) ;
5) Accepter ou refuser la solution x', en appliquant une certaine "règle d'acceptation" (généralement, la règle de Metropolis, décrite en Figure 2) ;
6) Sauver le meilleur point rencontré ;
7) Si l'"équilibre thermodynamique" du système à la température T est atteint,
Alors abaisser légèrement la température T ;
Sinon Aller à l'étape **3)** ;
8) Si le "système est figé" (par exemple, la température T est inférieure à une température seuil voisine de 0),
Alors Aller à l'étape **9)** ;
Sinon Aller à l'étape **3)** ;
9) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 3. Algorithme du recuit simulé.

3.1.2. *La méthode de la diffusion simulée [30]*

L'idée, dans cette méthode, est d'introduire des fluctuations aléatoires autorisant des dégradations de la fonction objectif tout en préservant la descente le long des gradients. Le minimum de la fonction f est localisé à partir du comportement asymptotique des solutions de l'équation différentielle ordinaire du gradient :

$$\dot{\mathbf{x}} = -\nabla f(\mathbf{x}) \tag{19}$$

pour laquelle le minimum est un état stable. Il existe, cependant, un risque majeur : celui de rester piégé dans un minimum local de f plutôt que de converger vers le minimum global $\mathbf{x}^*$. Afin d'éviter cette difficulté, une perturbation stochastique est ajoutée à l'équation considérée (19) qui s'écrit alors :

$$\mathrm{d}\mathbf{x}_t = -\nabla f(\mathbf{x}_t)\mathrm{d}t + \sigma(t)\mathrm{d}W_t \tag{20}$$

où $\{W_t, t \geq 0\}$ est un mouvement Brownien standard (c'est-à-dire un processus de "marche au hasard") pour un choix approprié du coefficient scalaire de diffusion $\sigma(t)$.

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser et évaluer la valeur de la fonction objectif f = f(**x**) ;

2) Perturber cette solution pour obtenir une nouvelle solution **x'** = **x** + **Δx** ;

3) Calculer Δf =f(**x'**) - f(**x**) ;

4) Si $\Delta f < E_c$
Alors accepter la nouvelle solution **x'** ; faire $\mathbf{x} \leftarrow \mathbf{x}'$ et $E_c \leftarrow E_c - \Delta f$;
Sinon refuser la solution **x'** ;

5) Sauver le meilleur point rencontré ;

6) Si l'"équilibre thermodynamique" du système est atteint,
Alors diminuer l'énergie cinétique E_c ;
Sinon Aller à l'étape **2)**;

7) Si l'énergie cinétique E_c est voisine de 0,
Alors Aller à l'étape **8)** ;
Sinon Aller à l'étape **2)** ;

8) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 4. Algorithme du recuit microcanonique.

On suppose que ∇f admet une constante de Lipschitz K, et satisfait à la limite de croissance :

$$|\nabla f(\mathbf{x})|^2 \leq K(1+ \| \mathbf{x} \|^2), \text{ pour une certaine constante } K, \text{ et pour tout } \mathbf{x} \in \Re^n. \tag{21}$$

Alors, pour $\sigma(t) = c/\sqrt{\log(t+2)}$, avec $c > 0$, on peut montrer que la distribution de probabilité $p(\mathbf{x}_t)$ converge vers une limite de densité de Gibbs proportionnelle à $\exp\left(-\frac{f(\mathbf{x})}{T}\right)$ lorsque la "température absolue" $T = \sigma^2(t) \to 0$ quand $t \to \infty$; cette densité limite est "concentrée" autour du minimum global $\mathbf{x}^*$ de f. D'autres choix de $\sigma(t)$ peuvent conduire à la convergence vers un minimum local de f avec une probabilité plus grande que vers le minimum global.

En fait, le résultat important suivant :

$$E\left(\| \mathbf{x}_t - \mathbf{x}^* \|^2 \log t\right) \geq \gamma \tag{22}$$

peut être établi pour une certaine valeur de $\gamma > 0$ et pour tout t suffisamment grand. Ceci fournit une borne inférieure pour la convergence au sens des moindres carrés de $\mathbf{x}_t$ vers $\mathbf{x}^*$.

Cette procédure utilisant les solutions $\mathbf{x}_t$ de (20) pour localiser $\mathbf{x}^*$ est aussi appelée "recuit stochastique" ou "gradient stochastique". En pratique, une méthode numérique est nécessaire pour résoudre l'équation différentielle stochastique (20).

3.1.3. *Le recuit microcanonique [21]*

Dans le cas où l'on considère le système isolé (c'est-à-dire qu'il n'a aucun échange de chaleur avec son environnement), une analyse microcanonique peut être faite : la principale propriété du système physique, dans ce cas, est que son énergie totale est constante, quelle que soit son évolution dynamique. Suivant cette analyse, Creutz a proposé une variante du recuit simulé, le "recuit microcanonique" [21]. L'énergie totale du système est conservée au cours du processus. L'énergie totale du système est la somme de l'énergie potentielle et de l'énergie cinétique :

$$E_{\text{totale}} = E_p + E_c. \tag{23}$$

L'énergie cinétique E_c joue un rôle similaire à celui de la température pour le recuit simulé ; elle est contrainte à être positive. E_c permet de retrancher ou d'ajouter de l'énergie au système selon la perturbation effectuée. L'énergie du problème à minimiser est alors l'énergie potentielle E_p. L'algorithme accepte toutes les perturbations vers des états d'énergie plus basse en ajoutant $-\Delta E$ (l'énergie potentielle perdue) à l'énergie cinétique E_c. Les mouvements vers des états de plus haute énergie sont acceptés seulement quand $\Delta E < E_c$, et l'énergie apportée sous forme d'énergie potentielle est retranchée de l'énergie cinétique. Ainsi, l'énergie totale demeure constante. L'algorithme est décrit sur la figure 4.

À chaque palier d'énergie, l'"équilibre thermodynamique" est atteint dès que le rapport $r_{éq} = \frac{\langle E_c \rangle}{\sigma(E_c)}$ de l'énergie cinétique moyenne observée sur l'écart-type de la distribution de E_c est "voisin" de 1.

L'équation (24) entre l'énergie cinétique et la température établit un lien entre le recuit simulé et le recuit microcanonique.

$$k_{\text{B}} T = \langle E_c \rangle \quad (k_{\text{B}} = \text{constante de Boltzmann}). \tag{24}$$

Cet algorithme possède plusieurs avantages par rapport au recuit simulé : il ne nécessite ni l'évaluation de fonctions transcendantes comme $\exp(\mathbf{x})$, ni le tirage de nombres aléatoires pour l'acceptation ou le refus d'une configuration. Il en résulte une rapidité plus grande. Néanmoins, Creutz a constaté que, dans le cas de systèmes de "petites tailles", la probabilité pour le système d'être piégé dans des états métastables est plus élevée [37].

3.1.4. *La méthode du seuil [6, 28]*

La méthode du seuil est une variante du recuit simulé. Elle a, jusqu'ici, été utilisée pour la résolution de problèmes d'optimisation combinatoire.

La principale différence entre les deux méthodes concerne les critères d'acceptation des solutions tentées : le recuit simulé accepte les configurations qui détériorent la fonction objectif f avec une certaine probabilité seulement, alors que la méthode du seuil accepte chaque nouvelle configuration, si la dégradation (éventuelle) de f ne dépasse pas un certain seuil T dépendant de l'itération k. L'algorithme est présenté sur la figure 5.

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser et évaluer la valeur de la fonction objectif f = f(**x**) ;
2) Choisir un seuil initial T ;
3) Perturber cette solution pour obtenir une nouvelle solution **x'** = **x** + **Δx** ;
4) Calculer Δf =f(**x'**) - f(**x**) ;
5) Si Δf < T,
Alors accepter la nouvelle solution **x'** ; faire **x** ← **x′** ;
6) Sauver le meilleur point rencontré ;
7) Si la qualité de l'optimum ne s'améliore pas depuis un "certain temps" ou si un nombre donné d'itérations a été atteint,
Alors abaisser le seuil T ;
8) Si le seuil T est proche de 0,
Alors aller à l'étape **10)** ;
9) Aller à l'étape **3)** ;
10) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 5. Algorithme de la méthode du seuil.

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser et évaluer la valeur de la fonction objectif f = f(**x**) ;
2) Initialiser la "quantité de pluie" UP > 0 ;
3) Initialiser le "niveau d'eau" WATER-LEVEL > 0 ;
4) Perturber cette solution pour obtenir une nouvelle solution **x'** = **x** + **Δx** ;
5) Evaluer la nouvelle valeur de f ;

6) Si f > WATER-LEVEL,
Alors accepter la nouvelle solution **x'** ; faire **x** ← **x′** ;
augmenter le niveau WATER-LEVEL de la quantité UP ;
7) Sauver le meilleur point rencontré ;

8) Si la fonction n'a pas été améliorée depuis longtemps ou s'il y a eu trop d'évaluations de fonctions,
Alors Aller à l'étape **9)** ;
Sinon Aller à l'étape **4)** ;

9) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 6. Algorithme de la méthode du grand déluge.

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser et évaluer la valeur de la fonction objectif f = f(**x**) ;

2) Initialiser "l'écart" autorisé DEVIATION > 0 ;

3) Evaluer le RECORD initial : RECORD = f(**x**) ;

4) Perturber cette solution pour obtenir une nouvelle solution **x'** = **x** + Δ**x** ;

5) Evaluer la nouvelle valeur de f ;

6) Si f > RECORD - DEVIATION,
Alors accepter la nouvelle solution ; faire **x** $\leftarrow$ **x'** ;

7) Sauver le meilleur point rencontré ;

8) Si f > RECORD,
Alors RECORD = f(**x**) ;

9) Si la fonction n'a pas été améliorée depuis longtemps ou s'il y a eu trop d'évaluations de fonctions,
Alors aller à l'étape **10)** ;
Sinon aller à l'étape **4)** ;

10) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 7. Algorithme de la méthode du voyage de record en record.

La méthode se compare favorablement au recuit simulé pour des problèmes d'optimisation combinatoire comme celui du voyageur de commerce (problème NP-difficile). Nous avons réalisé une adaptation de cette méthode à la résolution des problèmes à variables continues sur le modèle du recuit simulé continu.

3.1.5. *La méthode du "grand déluge" [29]*

L'heuristique que nous présentons maintenant ainsi que la suivante (Sect. 3.1.6) sont des méthodes de *maximisation* de la fonction objectif (un petit changement de la fonction initiale est donc nécessaire). Ce sont des variantes du recuit simulé et de la méthode du seuil. Les différences se situent au niveau des lois d'acceptation des solutions qui dégradent la fonction objectif. De plus, si la méthode du recuit simulé nécessite le choix délicat d'un certain nombre de paramètres, ces deux méthodes paraissent plus simples d'utilisation puisqu'elles comportent moins de paramètres (2 seulement). L'algorithme de la "méthode du grand déluge" se présente comme suit sur la figure 6.

L'allégorie du grand déluge permet de comprendre le mécanisme intuitif de cette méthode : pour garder les pieds au sec, le randonneur va visiter les points culminants de la région explorée. Alors que le niveau d'eau ne fait que monter, un inconvénient immédiat apparaît, celui de la séparation des "continents", qui devrait piéger l'algorithme dans des maximums locaux. Toutefois, l'auteur présente des résultats, sur des problèmes combinatoires, tout à fait comparables à ceux obtenus avec d'autres méthodes d'optimisation globale [29].

3.1.6. *La méthode du "voyage de record en record" [29]*

Cette autre variante, intitulée "voyage de record en record" est présentée sur la figure 7. Dans cette méthode, n'importe quelle solution peut être acceptée du moment qu'elle n'est pas "beaucoup plus mauvaise" que la meilleure valeur RECORD obtenue précédemment. On retrouve une certaine similitude avec la méthode précédente, la différence entre le RECORD et l'écart DEVIATION correspondant au niveau d'eau WATER-LEVEL.

Dans cette méthode, comme dans la précédente, il n'y a que deux paramètres à régler (la quantité d'eau UP pour l'une ou l'écart DEVIATION pour l'autre et le critère d'arrêt dans les deux cas). Le choix du premier paramètre est important, puisqu'il est un compromis entre la vitesse de convergence et la qualité du maximum obtenu.

L'auteur précise que les résultats de ces deux méthodes sur le problème du voyageur de commerce de dimension supérieure à 400 villes sont meilleurs que ceux obtenus avec le recuit simulé.

3.2. La méthode Tabou [25, 32, 34, 35]

La méthode de recherche Tabou ("Tabu Search"), mise au point par Glover, est une technique récente d'optimisation combinatoire. Plusieurs auteurs la présentent comme une alternative au recuit simulé. D'autres ont souligné l'intérêt d'une combinaison de la technique Tabou avec le recuit simulé ; Glover préconise ce type de combinaison, qu'il considère comme l'une des finalités de la stratégie Tabou.

Comme le recuit simulé, la méthode Tabou est conçue en vue de surmonter les minimums locaux de la fonction objectif. Nous en décrivons succinctement le principe.

À partir d'une configuration initiale quelconque, Tabou engendre une succession de configurations qui doit aboutir à la configuration optimale. À chaque itération, le mécanisme de passage d'une configuration, soit s, à la suivante, soit t, est le suivant :

- on construit l'ensemble des "voisins" de s, c'est-à-dire l'ensemble des configurations accessibles en un seul "mouvement" élémentaire à partir de s (si cet ensemble est trop vaste, on en extrait aléatoirement un sous-ensemble de taille fixée) : soit $V(s)$ l'ensemble (ou le sous-ensemble) envisagé ;
- on évalue la fonction objectif f du problème pour chacune des configurations appartenant à $V(s)$. La configuration t, qui succède à s dans la chaîne de

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser et évaluer la valeur de la fonction objectif f = f(**x**) ;

2) Faire n fois :

Perturber la solution **x** pour obtenir une nouvelle solution **x'** ;

(Si le mouvement de **x** à **x'** ∈ « Liste-tabou », Alors effectuer une autre perturbation de **x**)

Evaluer la valeur de f au point **x'** ;

Fin faire ;

3) Accepter le meilleur point **x'** parmi les n points voisins obtenus ; faire **x ←x'** ;

4) Mettre le mouvement inverse du mouvement effectué dans « Liste-tabou » (à la place du plus ancien élément de cette liste) ;

5) Sauver le meilleur point rencontré ;

6) Si pas d'amélioration depuis m itérations,

Alors Aller à l'étape **7)** ;

Sinon Aller en **2)** ;

7) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 8. Algorithme de la méthode Tabou simple.

Markov construite par Tabou, est la configuration de $V(s)$ en laquelle f prend sa valeur minimale.

Notons que la configuration t est adoptée même si $f(t) > f(s)$: c'est grâce à cette particularité que Tabou permet d'éviter les minimums locaux de f.

Cependant, telle quelle la procédure ne fonctionne généralement pas, car il y a un risque important de retourner à une configuration déjà retenue lors d'une itération précédente, ce qui provoque l'apparition d'un cycle. Pour éviter ce phénomène, on tient à jour, à chaque itération, une "liste tabou" de mouvements interdits ; cette liste – qui a donné son nom à la méthode – contient les mouvements inverses $(t \to s)$ des m derniers mouvements $(s \to t)$ effectués (typiquement, $m = 7$). La recherche du successeur de la configuration courante s est alors restreinte aux voisins de s qui peuvent être atteints sans utiliser un mouvement de la liste tabou. La procédure peut être stoppée dès que l'on a effectué un nombre donné d'itérations, sans améliorer la meilleure solution atteinte jusqu'ici.

L'algorithme ainsi décrit, représenté sur la figure 8, est dit "Tabou simple". Selon De Werra *et al.* [25], il serait plus efficace que le recuit simulé pour le problème

modèle du "coloriage d'un graphe". Cependant, le mode de construction de la liste tabou – qui, pour une simple raison d'économie de place mémoire, contient des *mouvements* interdits, et non des *configurations* interdites – peut bloquer l'accès à certaines solutions, pourtant non encore visitées. Pour éviter cet inconvénient, on peut employer la méthode plus complexe dite "Tabou généralisée", qui prévoit la possibilité d'annuler le statut tabou d'un mouvement, lorsque le bénéfice escompté est "suffisant" (cette circonstance est appréciée à l'aide de la notion de "niveau d'aspiration", qui est précisée en détail dans les articles référencés).

En outre, dans la description de la méthode que nous avons faite jusqu'ici, l'usage de la mémoire se limite à un contrôle *à court terme* du déroulement de l'exploration. Dans une version plus complète, le rôle de la mémoire influence également le processus de recherche *à long terme*, grâce à deux nouveaux concepts : l'"intensification" et la "diversification", brièvement présentés maintenant.

L'intensification consiste à interrompre – périodiquement et pour une durée limitée – le déroulement normal de l'algorithme, de façon à accentuer l'effort d'exploration dans certaines régions, identifiées comme particulièrement prometteuses. On peut, par exemple, retourner à l'une des meilleures solutions rencontrées jusqu'ici, puis reprendre l'exploration à partir de cette solution, en "fouillant" davantage l'espace des solutions : à cet effet, certains préconisent de réduire la longueur de la liste tabou utilisée ; d'autres élargissent le voisinage évalué à chaque itération ; d'autres encore font appel à un algorithme de descente locale – notamment la méthode du "simplex" de Nelder et Mead [55] –, qui permet d'affiner rapidement la solution analysée.

La diversification est le concept inverse de l'intensification. L'objectif visé est de rediriger la recherche vers des régions de l'espace où elle n'est pas encore allée (ou trop peu souvent), afin d'éviter de laisser de grandes régions totalement inexplorées. Une procédure élémentaire consiste à interrompre périodiquement le déroulement normal de l'algorithme, pour le reprendre à partir d'une nouvelle solution choisie au hasard. Gendreau *et al.* [31] ont proposé une "diversification continue", qui consiste à favoriser, tout au long de l'exploration, les caractéristiques rencontrées rarement. D'autres auteurs ont proposé de "relaxer" certaines contraintes d'admissibilité des solutions d'un problème, de manière à aplanir le paysage de l'espace des solutions et, de là, faciliter l'accès à des régions nouvelles.

Enfin, un sujet différent a fait l'objet, jusqu'ici, d'une poignée seulement de travaux de recherche : la "recherche Tabou continue" [4, 22, 40, 62, 63, 71]. Il s'agit de proposer des moyens d'adapter cette technique d'optimisation combinatoire aux problèmes à variables continues posés, par exemple, par l'optimisation des circuits. La difficulté principale se trouve dans l'adaptation du statut "tabou" d'un mouvement, ainsi que dans la définition du voisinage d'une solution. En effet, ces deux concepts sont élémentaires dans le cas discret, mais ne sont pas aisément transposables au cas continu. Les résultats publiés, relatifs à l'optimisation par des méthodes de type "Tabou simple" de fonctions analytiques de test, sont peu convaincants en comparaison du recuit simulé. L'adaptation au cas continu des notions complémentaires d'intensification et de diversification est en cours au sein de notre équipe : elle semble apporter une amélioration sensible des performances.

1) Choisir, au hasard, une population initiale composée de n éléments codés ;
2) Phase de reproduction : générer m fils à l'aide des opérateurs de croisement et de mutation ;
3) Evaluer f en chacun des individus ;
4) Phase de sélection : prendre les n meilleurs éléments parmi les m+n éléments (population initiale et fils) pour composer la génération suivante ;
5) Sauver le meilleur élément rencontré ;

6) Si le nombre de générations maximal n'est pas atteint,
Alors Aller en **2)** ;

7) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 9. Principe d'un algorithme génétique.

3.3. Les algorithmes génétiques [36, 38]

Les principes fondamentaux des algorithmes génétiques ont été exposés par Holland. Leur implémentation informatique a été expérimentée par Goldberg. Il s'agit d'une technique de recherche globale qui imite des opérateurs génétiques naturels. Des opérateurs inspirés par le mécanisme de la sélection naturelle (qui détermine quels membres d'une population survivent et se reproduisent) et de la reproduction sexuée (qui assure le brassage et la recombinaison des gènes parentaux, pour former des descendants aux potentialités nouvelles) sont appliqués à une population de tableaux binaires codant l'espace des paramètres.

Une population initiale de N individus est aléatoirement choisie, un individu correspond à une solution possible du problème posé. Les opérateurs génétiques sont appliqués à cette population afin de créer des enfants à partir de parents. La nouvelle population, appelée la génération suivante, est constituée en sélectionnant les N meilleurs individus. En itérant ce processus, on enrichit successivement la population avec des individus plus efficaces. À chaque génération, l'algorithme explore des domaines différents de l'espace des paramètres et dirige alors la recherche vers les régions où une haute probabilité de trouver une meilleure performance existe.

Les algorithmes génétiques convergent globalement à partir d'une population initiale déterminée aléatoirement. Ils sont intrinsèquement parallèles. En effet, toutes les chaînes ou individus dans une population évoluent simultanément sans coordination centrale. Pour réaliser leur plein potentiel, les algorithmes génétiques gagnent à être implémentés sur des architectures informatiques parallèles.

Les différentes étapes : population initiale, reproduction, et sélection sont données dans la figure 9.

Le principal opérateur agissant sur la population de parents est le croisement ou "crossover", qui est appliqué avec une certaine probabilité, appelée taux de croisement (typiquement, Pc = 80 %). Pour appliquer cet opérateur, deux chaînes de la population courante sont tirées au hasard et coupées entre deux bits aléatoirement choisis sur les chaînes. Les nouvelles chaînes sont alors créées en interchangeant les différentes parties de chaque chaîne : ceci est représenté par la figure 10 sur des variables (ou individus) codées sur 5 bits. Du fait du mécanisme expliqué plus loin, cet opérateur permet de diriger la recherche vers des régions de l'espace d'étude meilleures en utilisant la connaissance déjà présente dans la population courante.

Le second opérateur est la mutation. Il permet d'introduire de nouvelles informations dans la population. Cet opérateur est appliqué avec une certaine probabilité, appelée taux de mutation (typiquement, Pm = 5 à 10 %). Un lancé de dé est effectué pour chaque bit de la population courante afin de savoir si la mutation doit être exécutée sur le bit en question : ceci est représenté par la figure 11 sur un individu codé sur 4 bits.

Alors, la nouvelle génération est constituée par les meilleurs éléments sélectionnés dans l'ensemble "ancienne population et fils créés". Le nombre d'éléments de la nouvelle génération doit être égal à la taille de la population de départ.

Dans l'état actuel de la théorie des algorithmes génétiques, **il n'existe aucune garantie que la méthode découvre, en un temps fini, la solution optimale**. Les seuls résultats asymptotiques disponibles, obtenus à l'aide de la théorie des chaînes de Markov, garantissent l'obtention de l'optimum global au bout d'un nombre infini de générations (voir par exemple [64]).

En outre, le succès de la méthode dépend beaucoup du codage des individus. Cette importance du codage peut s'expliquer de la manière suivante. Une chaîne de bits appartient à toutes les "régions" que ses bits définissent. Par exemple, la chaîne 1101 appartient aux régions 11**, *1*1, *101, etc. : les * indiquent que la valeur du bit n'est pas spécifiée. Il en résulte qu'un algorithme génétique, qui manipule au total, typiquement, quelques milliers de chaînes, échantillonne en réalité un nombre bien supérieur de régions (en gros, le cube du nombre de chaînes [36] : c'est ce que Holland appelle le "parallélisme implicite").

Chaque région de l'espace des solutions est caractérisé par un "motif" (par exemple 11**), qui dépend du codage choisi ; ce motif possède un "ordre" (le nombre de bits spécifiés) et une "longueur" (la distance entre le premier et le dernier bit spécifié).

L'efficacité d'un algorithme génétique est fonction des motifs associés aux bonnes régions (*i.e.* aux régions qui contiennent une proportion élevée de "bonnes" solutions) : plus ces motifs sont courts et compacts, mieux l'algorithme fonctionne. En effet, lorsqu'une chaîne située dans une "bonne" région prend part à un croisement, le motif associé à cette région a peu de chances d'être coupé, s'il est court : le plus souvent, il est transmis au descendant, qui se retrouve alors dans la même région ; comme il s'agit d'une "bonne" région, le parent et son descendant ont de grandes chances d'appartenir à la génération suivante, et d'être sélectionnés pour la reproduction, ce qui amorce une réaction en chaîne.

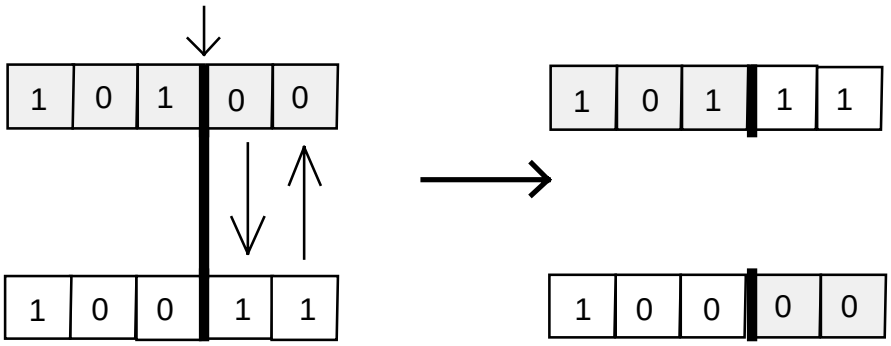

FIGURE 10. L'opérateur de croisement.

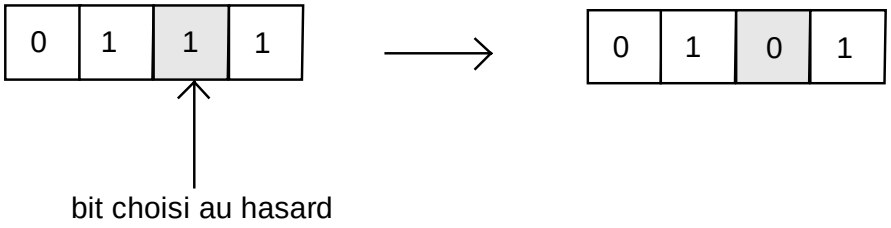

FIGURE 11. L'opérateur de mutation.

On montre que, d'une génération à la suivante, la répétition des croisements accroît exponentiellement le nombre de chaînes testées dans une "bonne" région, lorsque l'ordre et la longueur du motif de cette région sont "suffisamment" petits par rapport à la longueur des chaînes ("théorème fondamental" des algorithmes génétiques [36]).

En multipliant ainsi les chaînes testées dans les "bonnes" régions, l'algorithme accroît évidemment les chances de trouver la solution optimale recherchée, qui correspond au regroupement, dans une même chaîne, des "bons" motifs précédents, dénommés "blocs de construction".

Néanmoins, ces considérations théoriques ont des conséquences opérationnelles limitées : en pratique, le choix du codage relève encore plus souvent de l'art que de la science... Dans les problèmes combinatoires, le codage est souvent suggéré par la nature même du problème, ce qui induit des performances inégales pour les algorithmes génétiques.

Peu de travaux publiés jusqu'ici concernent l'optimisation de fonctions à n variables continues. Les plus connus sont ceux de Michalewicz *et al.*, qui ont développé les codes GENOCOP, GENOCOP II et GENOCOP III, pour l'optimisation globale avec contraintes (voir par exemple [50]). Le premier objectif est le choix d'un codage approprié, indépendant de l'application. Chaque "individu" est, ici, un vecteur à n composantes réelles : la chaîne de bits correspondante est construite

1) Engendrer, aléatoirement, une population de N points du domaine S ;
Calculer, pour chacun des points, la valeur de la fonction f ;
Calculer le total de ces valeurs dans la population ;

2) Faire N fois :
- Choisir, aléatoirement, deux points de la population courante ;
- Prendre comme « père » le meilleur des deux points ;
- Remplacer l'autre point par un « fils » du premier, engendré selon la « loi de filiation » courante (équation 25) ;
- Calculer la valeur de la fonction f associée au nouveau point, et mettre à jour le total de f dans la population ;

Fin faire ;

3) Si le total de f dans la population s'est amélioré,
Alors Aller à l'étape **2)** ;
Sinon Aller en **4** ;

4) Si critère d'arrêt atteint,
Alors Aller à l'étape **6)** ;
Sinon Aller en **5** ;

5) Ajuster la loi de filiation, en réduisant les paramètres d'échelle ;
Aller en **2** ;

6) Solution = meilleur point trouvé ; Arrêt du programme.

FIGURE 12. Algorithme de la méthode de recherche distribuée.

simplement par concaténation de n mots binaires respectivement associés aux n coordonnées. Se posent alors un problème de précision (paramètres réels représentés par des mots binaires), et surtout un problème de choix d'un code binaire (entiers représentés par des mots binaires).

3.4. AUTRES MÉTHODES

3.4.1. *La méthode de recherche distribuée [20]*

L'algorithme de "recherche distribuée" fait évoluer une distribution de probabilités de visite sur le domaine de recherche. Cette distribution converge vers un état où la densité de probabilité est maximale au voisinage des extrêmums recherchés, c'est-à-dire vers un état où la probabilité de visite est "concentrée" sur les zones où la fonction objectif prend les valeurs extrêmes recherchées.

L'idée de base est très simple : soient $\mathbf{x}_1$ et $\mathbf{x}_2$ deux points du domaine S de recherche tels que $f(\mathbf{x}_1) < f(\mathbf{x}_2)$, alors il existe un ensemble de points $\mathbf{x}$ dans un voisinage de $\mathbf{x}_1$ dont la taille varie au cours du processus et tel que $f(\mathbf{x}) < f(\mathbf{x}_2)$. Il s'agit alors pour l'algorithme d'échantillonner le domaine S

1) Choisir, aléatoirement, une solution initiale **x** du système à optimiser ;
2) Initialiser "l'amplitude du bruit" ;
3) Appliquer, à la fonction objectif "bruitée", une méthode de descente, à partir de la solution courante ;
4) Diminuer l'amplitude du bruit ;
5) Si l'amplitude du bruit est nulle,
Alors Aller à l'étape **6)** ;
Sinon Aller à l'étape **3)** ;
6) solution = meilleure solution rencontrée ; Arrêt du programme.

FIGURE 13. Algorithme de la méthode du bruitage.

suivant une distribution de probabilité, contrôlée par une estimation de la taille du voisinage adéquate selon les différentes étapes du processus.

La génération des points visités se fait indépendamment sur chaque coordonnée au moyen d'une fonction de génération, dite "loi de filiation", du type :

$$\mathbf{x}_i = s_i \mathrm{tg}(\pi u_i) + m_i, \tag{25}$$

où : u_i est un nombre aléatoire uniforme sur $]-1/2, +1/2[$, m_i sont les composantes du centre de la distribution, s_i sont les paramètres d'échelle, c'est-à-dire les paramètres qui définissent la taille du voisinage du minimum, ou encore les quartiles (*i.e.* les valeurs de $\mathbf{x}$ pour lesquelles la fonction de répartition vaut $1/4$ ou $3/4$).

La variable aléatoire ainsi définie obéit à une loi de Cauchy n-dimensionnelle de densité :

$$g(\mathbf{x}; m, S) = \Pi_{i=1}^{n} \left[\frac{1}{\pi s_i} \frac{1}{1 + \left(\frac{\mathbf{x}_i - m_i}{s_i} \right)^2} \right] . \tag{26}$$

La loi de Cauchy n'a pas de moments et sa variance est infinie (la densité décroît lentement et n'est jamais négligeable). D'après l'auteur, les variables de Cauchy ont des propriétés particulièrement favorables pour cette méthode.

Pratiquement, la méthode est mise en œuvre en suivant l'algorithme décrit dans la figure 12.

La probabilité de visite tend à devenir de plus en plus concentrée sur les zones où la fonction objectif prend ses meilleures valeurs. Aucune condition n'est nécessaire pour appliquer cet algorithme, il suffit de pouvoir évaluer la fonction objectif en n'importe quel point du domaine.

3.4.2. *La méthode du bruitage [15]*

Cette méthode toute récente est une heuristique qui se compare favorablement au recuit simulé sur certains problèmes d'optimisation combinatoire. Son adaptation aux problèmes à variables continues reste un sujet d'étude.

La méthode utilise un algorithme de descente, c'est-à-dire un algorithme qui, à partir d'une solution initiale, effectue des améliorations itératives jusqu'à atteindre un minimum local. Partant d'un point quelconque $\mathbf{x}$ du domaine S, les données sont "bruitées", ce qui signifie que les valeurs prises par la fonction f sont changées d'une certaine façon, puis l'algorithme de descente est appliqué en utilisant la fonction bruitée. À chaque itération, l'amplitude du bruitage de f diminue jusqu'à annulation complète. La meilleure solution rencontrée est considérée comme le minimum global. L'algorithme de cette méthode est proposé sur la figure 13.

3.4.3. *La méthode Aliénor [16–18]*

La méthode Aliénor, proposée par Cherruault *et al.*, repose sur une suite de transformations réductrices qui permet de ramener toute fonction de plusieurs variables à une fonction d'une seule variable : l'angle polaire d'une spirale d'Archimède. On peut alors utiliser, pour résoudre le problème multivariable, les méthodes puissantes habituellement mises en œuvre pour le cas unidimensionnel.

Par exemple, dans le cas d'un problème à 4 variables, remplaçons les composantes du vecteur $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ par :

$$\begin{array}{lll} x_1 = r_1 \cos\theta_1 & x_2 = r_1 \sin\theta_1 & r_1 = a_1\theta_1 \\ x_3 = r_2 \cos\theta_2 & x_4 = r_2 \sin\theta_2 & r_2 = a_2\theta_2, \end{array} \tag{27}$$

ce premier changement de variables donne :

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= f(a_1\theta_1 \cos\theta_1, a_1\theta_1 \sin\theta_1, a_2\theta_2 \cos\theta_2, a_2\theta_2 \sin\theta_2) \\ &= g(\theta_1, \theta_2). \end{aligned} \tag{28}$$

La fonction f de 4 variables est ramenée à une fonction g de 2 variables. Un second changement de variables :

$$\theta_1 = r\cos\theta,\ \theta_2 = r\sin\theta \text{ avec } r = a\theta \tag{29}$$

aboutit à une fonction d'une seule variable θ soit :

$$\begin{aligned} f(x_1, x_2, x_3, x_4) = f(&a_1 a\theta \cos\theta \cos(a\theta\cos\theta),\ a_1 a\theta\cos\theta \sin(a\theta\cos\theta), \\ &a_2 a\theta \sin\theta \cos(a\theta\sin\theta),\ a_2 a\theta \sin\theta \sin(a\theta\sin\theta)) \end{aligned} \tag{30}$$

$$f(x_1, x_2, x_3, x_4) = G(\theta). \tag{31}$$

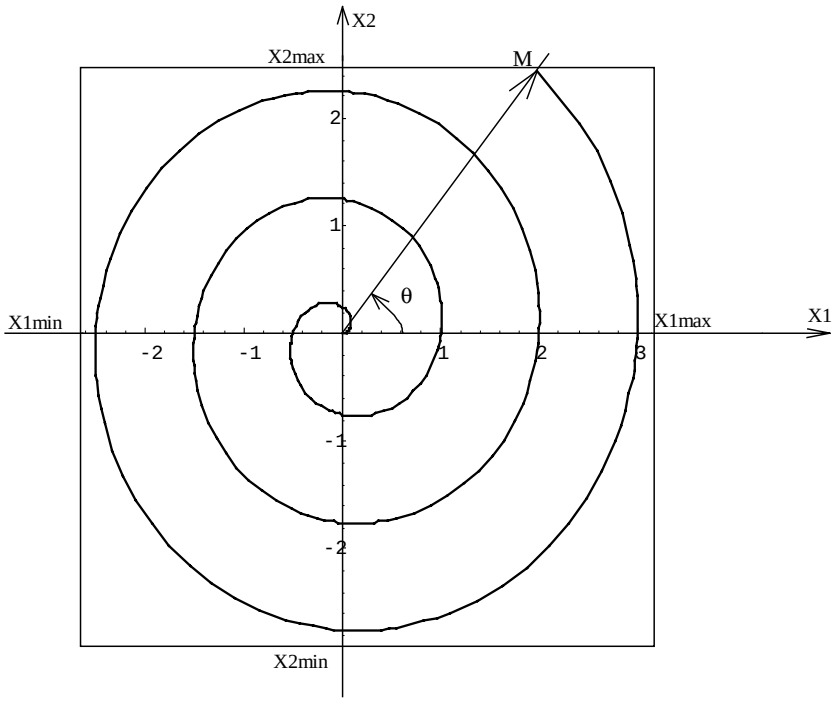

FIGURE 14. Exploration globale du plan paramétrée par θ (spirale d'Archimède).

Il est équivalent de chercher l'optimum global de f ou celui de G qui ne comporte plus qu'une seule variable θ. En effet, on montre que Min G tend vers Min f lorsque a, a_1, a_2 tendent vers 0 [18].

Ayant déterminé le minimum absolu de G, il suffit de remonter au minimum absolu de f par les transformations suivantes pour l'exemple à 4 variables :

$$\begin{aligned} x_1 &= a_1 a\theta \cos\theta \cos(\theta \cos\theta) \\ x_2 &= a_1 \theta \cos\theta \sin(\theta \cos\theta) \\ x_3 &= a_2 \theta \sin\theta \cos(\theta \sin\theta) \\ x_4 &= a_2 \theta \sin\theta \sin(\theta \sin\theta). \end{aligned} \tag{32}$$

La minimisation de $G(\theta)$ est faite suivant un algorithme classique de minimisation de fonction d'une seule variable.

Deux inconvénients apparaissent à l'utilisation de cette méthode :

- d'une part, le grand nombre de calculs de fonctions trigonométriques nécessaires dès que le nombre de variables augmente ;
- d'autre part, il est très difficile d'assurer simplement des contraintes du type :

$$a_j < x_j < b_j,\ 1 \leq j \leq n. \tag{33}$$

Il faut évaluer de nombreuses expressions du type :

$$x_j = a_j a\theta \cos\theta \cos(a\theta \cos\theta), \tag{34}$$

pour vérifier les inégalités précédentes.

La figure 14 représente l'exploration globale du plan (fonction objectif à 2 variables x_1, x_2) paramétrée par le paramètre unique θ.

La méthode Aliénor a permis de résoudre numériquement des équations aux dérivées partielles utilisées pour modéliser des systèmes biologiques, mais son efficacité n'est pas établie pour les problèmes à nombre de variables élevé. Les études que nous avons menées sur cette méthode et ses applications possibles dans le domaine de l'électronique nous ont permis de mettre en évidence cette difficulté face aux problèmes de grande dimension.

4. Conclusion

Cet état de l'art des méthodes d'optimisation globale, décrivant tant les principales approches "classiques" que les heuristiques récentes, permet de mesurer la première difficulté à laquelle est confronté l'utilisateur, face à un problème d'optimisation concret : celui du choix d'une méthode "efficace", capable de produire une solution "optimale" – ou de qualité acceptable – au prix d'un temps de calcul "raisonnable". Cette difficulté s'accroît encore quand il est nécessaire d'adapter les méthodes existantes, en particulier pour tenir compte de la nature mixte – combinatoire et continue – de nombre de problèmes posés.

Face à ce souci pragmatique de l'utilisateur, la théorie n'est pas d'un grand secours, car les théorèmes de convergence sont souvent inexistants, ou applicables sous des hypothèses très restrictives. En outre, le réglage "optimal" des multiples paramètres d'une heuristique, qui peut être préconisé par la théorie, est souvent inapplicable en pratique, car il induit un coût de calcul prohibitif. La comparaison systématique entre les différentes approches disponibles, lorsqu'elle est abordée dans la littérature, doit se limiter à des problèmes de test idéalisés.

Pour toutes ces raisons, le choix d'une "bonne" méthode fait généralement appel au savoir-faire et à l'"expérience" de l'utilisateur, plutôt qu'à l'application fidèle de règles bien établies. Mentionnons une nouvelle multiplication des possibilités, avec l'émergence des méthodes hybrides, qui s'efforcent de tirer avantage de la coopération de plusieurs méthodes, par exemple le recuit simulé pour localiser la "bonne vallée", suivi d'une descente de gradient pour affiner la solution. Une taxinomie des méta-heuristiques hybrides a été récemment proposée par Talbi [72],

pour tenter de guider le choix de l'utilisateur. Une voie de recherche prometteuse est aussi ouverte avec le développement de "systèmes multi-agents", qui seraient capables de puiser eux-mêmes dans la boîte des outils disponibles, de façon à améliorer la progression vers l'optimum, au gré des difficultés rencontrées...

Références

[1] E.H.L. Aarts et P.J.M. Van Laarhoven, *Simulated annealing: Theory and applications*. D. Reidel Publishing Company (1987).
[2] R.S. Anderssen, *Global optimization*, édité par R.S. Anderssen, L.S. Jennings et D.M. Ryan. Optimization, Univ. of Queensland Press, St Lucia (1972) 28-48.
[3] J.P. Barthélémy, G. Cohen et A. Lobstein, *Complexité algorithmique et problèmes de communication*. Masson, Collection CNET-ENST (1992).
[4] R. Battiti et G. Tecchiolli, The continuous reactive tabu search: Blending Combinatorial Optimization and Stochastic Search for Global Optimization. *Ann. Oper. Res.* **63** (1996) 53-188.
[5] R.W. Becker et G.V. Lago, A global optimization algorithm, dans *Proc. of the 8th Allerton Conference on Circuits and Systems Theory*. Montecillo, Illinois (1970) 3-12.
[6] M. Bertocchi et C.D. Odoardo, A stochastic algorithm for global optimization based on threshold accepting technique, dans *11th European Congress on Operational Research EURO XI*. Aachen, Germany (1991).
[7] G. Berthiau, *La méthode du recuit simulé pour la conception des circuits électroniques : adaptation et comparaison avec d'autres méthodes d'optimisation*. Thèse de Doctorat de l'École Centrale de Paris (1994).
[8] I.O. Bohachevsky, M.E. Johnson et M.L. Stein, Generalized Simulated Annealing for function optimization. *Technometrics* **28** (1986) 209-217.
[9] F.H. Branin et S.K. Hoo, A method for finding multiple extrema of a function of n variables, édité par F.A. Lootsma, *Numerical methods of nonlinear optimization*. Academic Press, London (1972).
[10] S.H. Brooks, A discussion of random methods for seeking maxima. *Oper. Res.* **6** (1958) 244-251.
[11] D.G. Brooks et W.A. Verdini, Computational experience with Generalized Simulated Annealing over continuous variables. *Amer. J. Math. Management Sci.* **8** (1988) 425-449.
[12] F. Catthoor, H. De Man et J. Vandewalle, SAMURAI: A general and efficient simulated annealing schedule with fully adaptive annealing parameters. *Integration, The VLSI Journal* **6** (1988) 147-178.
[13] V. Cerny, *Minimization of continuous functions by simulated annealing*, Internal Documentation HU-TFT-84-51. Research Institute for Theoretical Physics, University of Helsinki, Siltavuorenpenger 20c, SF-00170, Helsinki 17, Finland (1984).
[14] V. Cerny, Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *J. Optim. Theory Appl.* **45** (1985) 41-51.
[15] I. Charon et O. Hudry, *Le bruitage : une méthode prometteuse d'optimisation combinatoire*. ENST, Département d'Informatique, Rapport Interne Télécom Paris 92-D-005 (1992).
[16] Y. Cherruault et A. Guillez, Une méthode pour la recherche du minimum global d'une fonctionnelle. *C. R. Acad. Sci. Paris Sér. I Math.* **296** (1983) 175-178.
[17] Y. Cherruault, *Mathematical modelling in Biomedicine*. D. Reidel Publishing Company (1986).
[18] Y. Cherruault, A new method for global optimization (Alienor). *Kybernetes* **19** (1989) 19-32.
[19] A. Corana, M. Marchesi, C. Martini et S. Ridella, Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm. *ACM Trans. Math. Software* **13** (1987) 262-280.

[20] P. Courrieu, *Un algorithme de recherche distribuée pour l'optimisation difficile*. Univ. de Provence, Centre de Recherche en Psychologie Cognitive, Rapport Interne TF-9101 (1991).
[21] M. Creutz, Microcanonical Monte-Carlo simulation. *Phys. Rev. Lett.* **50** (1983) 1411-1414.
[22] D. Cvijovic et J. Klinowski, Taboo search. An approach to the Multiple Minima Problem. *Science* **667** (1995) 664-666.
[23] A. Dekkers et E.H.L. Aarts, Global optimization and simulated annealing. *Math. Programming* **50** (1991) 367-393.
[24] L. Devroye, Progressive global random search of continuous functions. *Math. Programming* **15** (1978) 330-342.
[25] D. De Werra et A. Hertz, Tabu search techniques: A tutorial and an application to neural networks. *OR Spektrum* **11** (1989) 131-141.
[26] L.C.W. Dixon et G.P. Szegö, *Towards global optimization*. North Holland, Amsterdam (1975).
[27] L.C.W. Dixon et G.P. Szegö, *Towards global optimization 2*. North Holland, Amsterdam (1978).
[28] G. Dueck et T. Scheuer, *Threshold accepting*. IBM Zentrum Heidelberg, Germany (1989).
[29] G. Dueck, New optimization heuristics, the great deluge and the record-to record travel. *J. Comput. Phys.* **104** (1993) 86-92.
[30] S. Geman et C.R. Hwang, Diffusions for global optimization. *SIAM J. Control Optim.* **24** (1986) 1031-1043.
[31] M. Gendreau, A. Hertz et G. Laporte, A Tabu Search Algorithm for the Vehicle Routing Problem. *Management Sci.* **40** (1994) 1276-1290.
[32] F. Glover, Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13** (1986) 533-549.
[33] F. Glover et H.J. Greenberg, New approaches for heuristic search: A bilateral linkage with artificial intelligence. *Eur. J. Oper. Res.* **39** (1989) 119-130.
[34] F. Glover, *Tabu search fundamentals and uses*, Working paper. Graduate School of Business, Box 419, University of Colorado, Boulder, CO (1995).
[35] F. Glover et M. Laguna, *Tabu search*. Kluwer Academic Publishers (1997).
[36] D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading (1989).
[37] L. Hérault, *Réseaux de neurones récursifs pour l'optimisation combinatoire ; Application à la théorie des graphes et à la vision par ordinateur*, Thèse de Doctorat de l'Institut National Polytechnique de Grenoble. INPG, Grenoble (1989).
[38] J.H. Holland, *Adaptation in natural and artificial systems*. Univ. of Michigan Press, Ann Arbor (1975).
[39] R. Horst, P.M. Pardalos, *Handbook of Global Optimization*. Kluwer Academic Publishers (1995).
[40] N. Hu, Tabu Search method with random moves for globally optimal design. *Int. J. Numer. Meth. Eng.* **35** (1992) 1055-1070.
[41] R.B. Kearfott, *Test results for an interval branch and bound algorithm for equality-constrained optimization*, édité par C. Floudas et P.M. Pardalos, *State of the Art in Global Optimization: Computational Methods and Applications*. Kluwer, Dordrecht, Netherlands (1996) 181-200.
[42] R.B. Kearfott et V. Kreinovich, *Applications of Interval Computations*. Kluwer, Dordrecht, Netherlands, *Applied Optimization* (1996).
[43] S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi, *Optimization by simulated annealing*, Research Report RC 9355. IBM, Yorktown Heights, NY (1982).
[44] S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi, Optimization by simulated annealing. *Science* **220** (1983) 671-680.
[45] A.V. Levy et S. Gomez, *The tunneling algorithm for the global optimization problem of constrained functions*, Technical Report 231. Univ. Nat. Auton. de Mexico (1980).

[46] A.V. Levy et S. Gomez, *The tunneling method applied to global optimization*, édité par P.T. Boggs, R.H. Byrd et R.B. Schnanel, Numerical Optimization 1984. SIAM Philadelphia (1984).
[47] A.V. Levy et A. Montalvo, The tunneling algorithm for the global minimization of functions. *SIAM J. Sci. Stat. Comp.* **6** (1985) 15-29.
[48] M. Lundy et A. Mees, Convergence of an annealing algorithm. *Math. Programming* **34** (1986) 111-124.
[49] N. Metropolis, A.R. Rosenbluth, M.N. Rosenbluth, A. Teller et E. Teller, Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21** (1953).
[50] Z. Michalewicz, *Genetic algorithms + Data structures = Evolution Programs*. Springer (1996).
[51] M. Minoux, *Programmation mathématique – Théorie et algorithmes*. Édition Dunod (1983).
[52] R.E. Moore, On computing the range of values of a rational function of n variables over a bounded region. *Computing* **16** (1976) 1-15.
[53] R.E. Moore, *Methods and applications of interval analysis*. SIAM, Philadelphia (1979).
[54] I. Mrad, *La méthode du recuit simulé pour la synthèse automatique d'un schéma électrique équivalent. Application à la modélisation de composant et à l'adaptation à large bande*. Thèse de Doctorat de l'École Centrale de Paris (1997).
[55] J.A. Nelder et R. Mead, A simplex method for function minimization. *Comput. J.* **7** (1965) 308-313.
[56] C. Poivey, *Méthodes d'optimisation globales pour la C.A.O. de circuits intégrés. Interface avec le simulateur SPICE-PAC*. Thèse de Doctorat de l'Université de Clermont-Ferrand (1988).
[57] C.R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. Mc Graw-Hill, *Advanced Topics in Comput. Sci. Ser.* (1995).
[58] A.H.G. Rinnooy Kan et G.T. Timmer, Stochastic methods for global optimization. *Amer. J. Math. Management Sci.* **4** (1984) 7-40.
[59] A.H.G. Rinnooy Kan et G.T. Timmer, *Global optimization*, Report 8612/A. Erasmus Univ. Rotterdam (1986).
[60] A.H.G. Rinnooy Kan et G.T. Timmer, Stochastic global optimization methods. Part I: Clustering methods. *Math. Programming* **39** (1987) 27-56.
[61] A.H.G. Rinnooy Kan et G.T. Timmer, Stochastic global optimization methods. Part II: Multi-level methods. *Math. Programming* **39** (1987) 57-78.
[62] E. Rolland, *A Tabu Search Method for Constrained Real-Number Search: Applications to Portfolio Selection*, Working Paper. The A. Gary Anderson Graduate School of Management, University of California, Riverside (1996).
[63] E. Rolland et H. Johnson, *Skewness and the Mean-Variance Frontier: A Tabu Search Approach*, Working Paper. The A. Gary Anderson Graduate School of Management, University of California, Riverside (1996).
[64] G. Rudolph, Convergence analysis of canonical genetic algorithms. *IEEE Trans. Neural Networks* **5** (1994) 96-101.
[65] K. Schittkowski et W. Hock, *Test examples for nonlinear programing codes*. Springer-Verlag, *Lecture Notes in Econom. and Math. Systems* **187** (1981).
[66] K. Schittkowski et W. Hock, *More test examples for nonlinear programing codes*. Springer-Verlag, *Lecture Notes in Econom. and Math. Systems* **282** (1988).
[67] P. Siarry, *La méthode du recuit simulé : application à la conception de circuits électroniques*. Thèse de Doctorat de l'Université Pierre et Marie Curie, Paris 6 (1986).
[68] P. Siarry et G. Dreyfus, *La méthode du recuit simulé : théorie et applications*. Éditeur IDSET (1988).
[69] P. Siarry, *La méthode du recuit simulé en électronique : adaptation et accélération. Comparaison avec d'autres méthodes d'optimisation. Application dans d'autres domaines*, Rapport d'habilitation à diriger les recherches en sciences. Université de Paris Sud, Centre d'Orsay (1994).

[70] P. Siarry, G. Berthiau, F. Durbin et J. Haussy, Enhanced Simulated Annealing for globally minimizing functions of many-continuous variables. *ACM Trans. Math. Software* **23** (1997) 209-228.

[71] P. Siarry et G. Berthiau, Fitting of Tabu Search to optimize functions of continuous variables. *Int. J. Numer. Methods Eng.* **40** (1997) 2449-2457.

[72] E.G. Talbi, *A taxonomy of hybrid meta-heuristics*, Rapport AS-183 du Laboratoire d'Informatique Fondamentale de Lille. Université des Sciences et Technologies de Lille (1998).

[73] A. Törn, *A search clustering approach to global optimization*, édité par L.C.W. Dixon et G.P. Szegö. North Holland, Amsterdam, *Towards Global Optimization* **2** (1978).

[74] A. Törn et A. Zilinskas, *Global optimization*, édité par G. Goos et J. Hartmanis. Springer Verlag, No. 350 (1989).

[75] D. Vanderbilt et S.G. Louïe, A Monte-Carlo simulated annealing approach to optimization over continuous variables. *J. Comput. Phys.* **56** (1984) 259-271.