Chapitre 3

Méthodes d'optimisation sans dérivées

Dans ce chapitre, nous nous intéressons aux méthodes d'optimisation sans dérivées. Ces méthodes sont développées pour résoudre des problèmes mathématiques de la forme :

minimiser une fonction objectif f,
éventuellement sujet à des contraintes imposant aux variables x d'appartenir à $\Omega \subseteq \mathbb{R}^n$,
et où les dérivées du problème ne sont pas accessibles.

Les algorithmes présentés sont classés dans trois catégories. La section 3.1 reprend des méthodes heuristiques. Il s'agit de méthodes qui n'ont pas de critère d'arrêt qui permet de prouver mathématiquement la qualité des solutions trouvées. La section 3.3 reprend des méthodes de recherche directe, tandis que la section 3.2 présente des méthodes basées sur des modèles. La dernière section revient sur des algorithmes d'optimisation sans dérivées qui utilisent des méthodes d'analyse de sensibilité.

3.1 Méthodes heuristiques

Bien que ces méthodes ne soient pas supportées par des résultats de convergence prouvés mathématiquement, ces méthodes sont fortement utilisées en pratique. En effet, elles sont souvent plus simple à comprendre et à implémenter que d'autres méthodes d'optimisation. De plus, elles peuvent être modifiées afin de mieux correspondre au problème et offrent, en général, de bonnes performances. Ces méthodes ont un intérêt supplémentaire, elles peuvent être utilisées comme sous-méthodes dans d'autres algorithme qui eux garantissent des résultats de convergence. Dans ce document, on présente la famille des *algorithmes génétiques* et l'algorithme de *Nelder-Mead*.

3.1.1 Algorithm *Hit-and-run*

Décrit dans [51, 16, 57], l'algorithme *Hit-and-run* est d'une conception assez simple. A chaque itération, la méthode compare l'itéré courant x à un autre candidat généré aléatoirement. Si ce candidat améliore la solution, alors l'itéré est mis à jour. La génération aléatoire des candidats se fait en deux étapes. Dans un premier temps, on génère une direction d à partir d'une distribution dense dans la sphère unité. Ensuite, la longueur du pas s est générée à partir d'une distribution uniforme, telle que $x + sd$ soit réalisable. Bien qu'une analyse de convergence de cet algorithme soit proposé dans [13], on le place dans la catégorie des heuristiques car il a un comportement fortement aléatoire et est d'une conception assez simple.

3.1.2 Algorithme du recuit simulé

L'algorithme du recuit simulé s'inspire d'une métaphore industrielle. D'une manière similaire à l'algorithme précédent, l'algorithme du recuit simulé génère des points à chaque itération. Par contre, un nouveau point généré $\hat{x}$ sera accepté avec une probabilité

$$P(\hat{x}|x_k) = \begin{cases} e^{\left(-\frac{f(\hat{x})-f(x_k)}{T_k}\right)} & \text{si } f(\hat{x}) > f(x_k), \\ 1 & \text{si } f(\hat{x}) \leq f(x_k), \end{cases}$$

où x_k est l'itéré courant et T_k un paramètre appelé température. Le paramètre T_k est mis à jour à chaque itération et la suite $\{T_k\}$ tend vers 0. Cela permet d'accepter des points moins bon que l'itéré courant afin de s'échapper d'un minimum local. Cet algorithme n'offre aucune garantie quant à la possibilité de trouver une bonne solution en un nombre fini d'itérations.

3.1.3 Algorithme génétique

Les algorithmes génétiques, aussi appelés algorithmes évolutifs, se basent sur la théorie de survie des individus les mieux adaptés de l'évolution. Pour continuer avec l'analogie biologique, on considère qu'une solution est individu, qu'un ensemble de solutions forme une population et que les individus se combinent pour en créer de nouveau. Le cadre général des algorithmes génétiques est décrit à l'algorithme 1. On remarque que celui-ci laisse beaucoup de flexibilité et d'interprétation, notamment au point 3 où on ne donne pas de critère d'arrêt, ce qui correspond à une heuristique. De même, la sélection de la population initiale, la métrique d'aptitude, la sélection des parents ou des survivants, la création d'un progéniture ou la mutation de celle-ci sont des étapes clefs dont on ne présente pas de méthode claire. On peut donner des bonnes pratiques pour certaines de ces décisions sans toutefois donner de règles fixes.

Algorithme 1 Algorithme génétique

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et une population initiale $P^0 = \{x^1, x^2, \ldots, x^{\bar{p}}\}$

0. Initialisation

$\gamma \in (0; 1)$	Probabilité de mutation
$k \leftarrow 0$	Compteur d'itération

1. aptitude

- ▷ Utiliser la valeur de la fonction objectif $f(x)$ pour déterminer une aptitude de survie de chaque individu $x \in P^k$

2. Croisement

- ▷ 2.1 Sélection : sélectionner 2 parents de la population P^k et aller à 2.2 ou sélectionner un survivant de P^k et aller à 2.4 ;
- ▷ 2.2 Croisement : utiliser les deux parents pour créer une progéniture ;
- ▷ 2.3 Mutation : avec une probabilité γ appliquer une mutation à la progéniture et vérifier la faisabilité de la mutation :
  - ▷ Si la mutation n'est pas réalisable, déclarer la progéniture morte et aller à 2.1
  - ▷ Sinon, déclarer la progéniture survivante et aller à 2.4 ;
- ▷ 2.4 Mise à jour de la nouvelle génération :
  - ▷ Placer le survivant dans la population P^{k+1},
  - ▷ Si $|P^{k+1}| \geq \bar{p}$, aller à 3
  - ▷ Sinon, aller à 2.1

3. Mise à jour

- ▷ Incrémenter $k \leftarrow k + 1$, stop ou aller à 1

C'est dans la flexibilité des algorithmes génétiques que réside leur force, mais aussi leur principal défaut. En effet, ceux-ci peuvent être adaptés à chaque problème particulier afin d'améliorer leur performance, mais on ne peut garantir que deux algorithmes génétiques aient un comportement similaire face à un même problème d'optimisation.

La qualité d'un individu est appelée son aptitude. Bien que celle-ci doit être liée à la valeur de la fonction objectif, plusieurs façons de calculer cette aptitude existe. Elles doivent toutefois respecter deux règles.

1. Pour un problème de minimisation d'une fonction f, si $f(x) < f(y)$, alors la valeur de l'aptitude de x doit être plus grande que celle de y. Cette règle s'inverse pour un problème de maximisation.
2. L'aptitude d'un individu doit être strictement positive.

On propose deux méthodes pour calculer une métrique d'aptitude utilisant la fonction objectif, il s'agit du l'aptitude de ranget de l'aptitude de valeur de la fonction.

— La aptitude de rang se base sur l'ordonnancement des points. On considère une population $P^k = \{x^1, x^2, \ldots, x^{\bar{p}}\}$ et les valeurs de la fonction correspondantes $\{f(x^1), f(x^2), \ldots, f(x^{\bar{p}})\}$. L'aptitude de rang d'un point x^i est le nombre m de points x^j, $j \in \{1, 2, \ldots, m\}$ tels que $f(x^i) \leq f(x^j)$. Il s'agit donc du nombre de points qui aboutissent à une plus petite évaluation de l'objectif que lui. Il faut noter que pour utiliser cette métrique, une liste de points de *mauvaise qualité* doit être fournie, sinon le plus mauvais point pourrait avoir une aptitude de 0 or celle-ci doit être strictement positive. Cette métrique est assez facile à comprendre et à utiliser mais a une certaine étrangeté. En effet, dès que les points de la population sont ordonnés, la valeur de la fonction objectif peut être effacée. Cela peut paraître bizarre pour une méthode d'optimisation.

— La aptitude de valeur de la fonction exploite directement l'évaluation de l'objectif. On considère également une population $P^k = \{x^1, x^2, \ldots, x^{\bar{p}}\}$ et les valeurs de la fonction correspondantes $\{f(x^1), f(x^2), \ldots, f(x^{\bar{p}})\}$. La aptitude de valeur de la fonction d'un point s'obtient en multipliant la valeur de la fonction correspondante par -1 et en déplaçant les valeurs telles que la plus petite aptitude de valeur de la fonction soit égale à 1. On peut la décrire comme

$$Fit(x^i) = -f(x^i) + \bar{f} + 1 \qquad \text{où } \bar{f} = \max_{i=1,2,\ldots,\bar{p}} f(x^i).$$

Cette métrique a l'avantage d'utiliser uniquement la population et de garder une certaine échelle entre les points. En effet, un point qui est nettement meilleur que les autres, aura une aptitude bien plus grande que les autres.

Une fois que la métrique d'aptitude est déterminée, il faut établir une méthode de sélection. Cette méthode doit permettre de choisir un individu survivant ou deux individus parents d'une population. Sélectionner deux parents revient à choisir deux fois un survivant, à l'exception que les deux survivants doivent être différents. Parmi les méthodes de sélection populaires, on présente la sélection élitiste, la roulette et le tournoi.

— **Sélection élitiste :** la stratégie élitiste consiste à choisir l'individu avec la meilleure aptitude, en choisissant les égalités aléatoirement avec une distribution uniforme. La stratégie élitiste appliquée m fois revient à sélectionner les m individus avec la meilleure aptitude. La sélection élitiste permet de s'assurer de garder la meilleure solution connue. Cette stratégie est la plus simple mais également extrêmement importante. Sélectionner un individu avec la stratégie élitiste permet d'obtenir quelques résultats de convergence.

— **Sélection par roulette :** on considère une population d'individus $\{x^1, x^2, \ldots, x^{\bar{p}}\}$ et leur aptitude correspondante $\{f^1, f^2, \ldots, f^{\bar{p}}\}$. La sélection par roulette choisit un individu aléatoirement avec les probabilités

$$prob(x^i) = f^i / \sum_{j=1}^{\bar{p}} f^j.$$

Notons que, puisque $f^i > 0$, la sélection par roulette donne des probabilités $prob(x_i) > 0$ et que

$$\sum_{i=1}^{\bar{p}} prob(x^i) = \sum_{i=1}^{\bar{p}} f^i / \sum_{j=1}^{\bar{p}} f^j = 1,$$

ce qui donne une distribution valide.

— **Sélection par tournoi :** la sélection par tournoi est sans doute celle qui se veut le plus proche de la métaphore biologique. L'idée est la suivante. A partir de la population entière, on sélectionne une sous-population. Celle-ci représente la partie de la population qui peut interagir entre elle. Ensuite, on peut appliquer une stratégie de sélectionner à partir de la sous-population. En comparaison avec le monde biologique, on peut utiliser deux fois d'affilée la même sous-population. Pour appliquer cette stratégie, on a besoin d'une population d'individus $\{x^1, x^2, \ldots, x^{\bar{p}}\}$ et de leur aptitude correspondante $\{f^1, f^2, \ldots, f^{\bar{p}}\}$, ainsi que d'une taille de tournoi $1 < N < \bar{p}$. La sélection par tournoi choisit d'abord N individus de la population

$\{x^1, x^2, \ldots, x^{\bar{p}}\}$ avec des probabilités égales. Le survivant ou le parent est ensuite choisi avec une stratégie élitiste ou une sélection par roulette parmi la sous-population.

Une fois la sélection effectuée, viennent les étapes de croisement et de mutation. Celles-ci se basent sur un système d'encodage. Chaque individu est représenté par un ensemble de *bits*, souvent appelé chromosomes pour préserver la métaphore. Plusieurs stratégies de croisement existent et se basent sur le fait de garder une partie des chromosomes de chaque parent pour créer un nouvel individu. Les stratégies de mutation consistent en l'inversion de certains *bits* choisis aléatoirement.

A la troisième étape de l'algorithme 1, l'algorithme génétique a la possibilité de s'arrêter. Toutefois, cet algorithme n'utilise pas de mesure intrinsèque pour mesurer la convergence de la méthode. Les critères d'arrêt les plus utilisés se basent donc sur un budget d'évaluation ou sur un nombre maximum d'itération sans amélioration de la valeur de l'objectif. A l'aide de certaines hypothèses sur la sélection et l'encodage, on peut trouver un résultat de convergence de l'algorithme génétique.

Théorème 3.1 (Convergence de l'algorithme génétique ([10, section 4.5], notre traduction)) *Supposons qu'un algorithme génétique est utilisé pour trouver* $\min\{f(x) : x \in \Omega\}$ *de telle manière que*

1. *la meilleure solution de la population* P^{k+1} *est au moins aussi bonne que la meilleure solution de la population* P^k*, i.e.*

$$\min\{f(x^i) : x^i \in P^{k+1}\} \leq \min\{f(x^i) : x^i \in P^k\};$$

2. *l'ensemble des points encodés est fini et il existe* $\epsilon > 0$ *tel que*

$$prob(c \in P^k) \geq \epsilon \qquad \text{pour n'importe quel point encodé } c \text{ et n'importe quelle itération } k.$$

Soient

$$f^* = \min_x\{f(x) : x \in \Omega\}, \qquad X^* = \operatorname*{argmin}_x\{f(x) : x \in \Omega\}$$
$$f^k_{best} = \min_x\{f(x^i) : x^i \in P^k\} et \qquad x^k_{best} = \operatorname*{argmin}_x\{f(x^i) : x^i \in P^k\}.$$

1. *Si les points sont encodés de manière binaire, alors*

$$prob\left(\lim_{k\to\infty} f^k_{best} = f^*\right) = 1 \qquad et \qquad \lim_{k\to\infty} prob\left(P^k \cap X^* \neq \emptyset\right) = 1.$$

2. *Si les points sont encodés de manière continue,* Ω *est compact et* $f \in \mathcal{C}^0$*, alors étant donné une sous-suite convergente de* x^k_{best} *(*$x^{ki}_{best} \subset \left\{x^k_{best}\right\}$ *avec* $x^{ki}_{best} \to \bar{x}$*)*

$$prob\left(dist(x^k_{best}, X^*) \to 0\right) = 1.$$

On note que des méthodes pour satisfaire les hypothèse sont données dans ([10] section 4.2-4.5). Concrètement, les résultats de convergence de l'algorithme génétique peuvent se résumer à *"en essayant suffisamment de points, on finira par trouver la meilleure solution"*. Néanmoins, les algorithmes génétiques sont très populaires et offrent parfois de bonnes performances. D'une manière générale, si on trouve des stratégies d'encodage et de croisement qui correspondent très bien au problème, les algorithmes génétiques auront de bonnes performances ; si les stratégies d'encodage et de croisement ne correspondent pas vraiment au problème, alors il sera plus intéressant d'utiliser d'autres méthodes.

3.1.4 Algorithme d'essaim de particules

L'algorithme d'essaim de particules est également un algorithme basée sur un ensemble de points, la population. A chaque élément de la population est associé une vitesse. A chaque itération, la population est mise à jour selon une série de règles basées sur un certain nombre de paramètres et des poids aléatoires. Un avantage de cette méthode est son aspect de recherche globale qui devrait éviter les minima locaux.

3.1.5 Algorithme de Nelder-Mead

L'algorithme de Nelder-Mead est une des méthodes d'optimisation sans dérivées les plus populaires. Celle-ci a été introduite sous le nom de méthode du simplexe [48]. Sa popularité vient à la fois de sa simplicité d'implémentation et des performances. L'algorithme original de possédait pas de résultats de convergence, même sur des fonctions différentiables et convexes [42], mais des versions modifiées de l'algorithme existent et assurent sa convergence vers un point stationnaire [23, 37, 61].

Comme son nom original l'indique, la méthode de Nelder-Mead se base sur un simplexe. Dans un espace de dimension n, un simplexe est le sous espace défini comme l'enveloppe convexe des $(n+1)$ points formant ses sommets.

A chaque itération k, on considère l'ensemble des $(n+1)$ sommets formant le simplexe $Y_k = \{y_k^0, y_k^1, \ldots, y_k^n\}$, ordonnés par ordre croissant de la valeur de f. Ensuite une opération de réflexion, d'expansion et/ou de contraction est appliquée pour remplacer le dernier sommet. Le nouveau sommet est

$$y = y^c + \delta(y^c - y^n),$$

où $\delta \in \mathbb{R}$ et $y^c = \sum_{i=0}^{n-1} y^i/n$ est le centroïde des n premiers sommets. La valeur du paramètre δ définit l'opération appliquée et celle-ci dépend du succès des transformations précédentes. On peut également appliquer une réduction qui ne garde que le meilleur sommets et en remplaçant tous les autres de la manière suivante :

$$y^{s_i} = y^0 + \delta^s(y^i - y^0), \quad i = 1, 2, \ldots, n.$$

On note δ^s, δ^{ic}, δ^{oc}, δ^r, δ^e le paramètre δ pour les opérations de rétrécissement, de contraction interne, de contraction externe, de réflexion et d'expansion respectivement. Une description de l'algorithme de Nelder-Mead est disponible à l'algorithme 2.

3.2 Méthodes basées sur des modèles

Comme leur nom l'indique, les méthodes d'optimisation sans dérivées basées sur des modèles utilisent des modèles de la fonction objectif pour optimiser celle-ci. L'idée derrière ces méthodes est d'utiliser ou de construire un modèle plus simple, que l'on peut optimiser plus facilement que le problème orignal. Si la qualité du modèle est satisfaisante, les solutions obtenues suite à l'optimisation du modèle devraient être des solutions intéressantes du problème orignal.

Il existe deux grandes catégories de modèle : les modèles statiques ou dynamiques. Les premiers sont plus souvent appelés substitut ou *surrogate*. Il s'agit de modèles qui ont un comportement similaires à la fonction objectif mais qui sont plus simples et plus rapides à évaluer. Ceux-ci sont en général fourni par l'utilisateur et restent les mêmes au cours de l'optimisation. L'utilisation la plus simple de ces modèles consistent à les optimiser en supposant que la solution du modèle sera une bonne solution du problème original.

Les modèles dynamiques sont construits et mis à jour au cours de l'optimisation. Par exemple, on présente le modèle quadratique. Celui-ci est construit à partir de la base des polynômes de degré inférieur ou égal à 2, qui possède $q+1 = \frac{(n-1)(n-2)}{2}$ éléments :

$$\begin{aligned}\phi(x) &= (\phi_0(x), \phi_1(x), \ldots, \phi_q(x)) \\ &= \left(1, x_1, x_2, \ldots, x_n, \frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2}, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n\right).\end{aligned}$$

Le modèle quadratique de la fonction objectif m_f sera donc défini par $\alpha \in \mathbb{R}^{q+1}$, $m_f(x) = \alpha^\top \phi(x)$.

La construction du modèle se fait à partir d'un ensemble de $p+1$ points de $\mathbb{R}^n$, $Y = \{y^0, y^1, \ldots, y^p\}$. Puisqu'on veut que le modèle soit le plus proche possible de la fonction objectif, on cherche à trouver $\alpha \in \mathbb{R}^{q+1}$ tel que $\sum_{y \in Y}(f(y) - m_f(y))^2$ soit minimal. En définissant une matrice $M(\phi, Y) \in$

Algorithme 2 Algorithme de Nelder-Mead

0. Initialisation

- ▷ Evaluer f aux points de $Y_0 = \{y_0^0, y_0^1, \ldots, y_0^n\}$
- ▷ Choisir les paramètres $0 < \delta^s < 1, -1 < \delta^{ic} < 0 < \delta^{oc} < \delta^r < \delta^e$
- ▷ $k \leftarrow 0$

1. Itération

- ▷ **Ordonnancement**
  - ▷ Ordonner les points de Y_k tels que $f^0 = f(y_k^0) \leq f^1 = f(y_k^1) \leq \cdots \leq f^n = f(y_k^n)$
- ▷ **Réflexion**
  - ▷ Construire y^c et $y^r = y^c + \delta^r(y^c - y^n)$
  - ▷ Evaluer $f^r = f(y^r)$
  - ▷ Si $(f^0 \leq f^r < f^{n-1})$
    - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^r\}$
    - ▷ $k \leftarrow k + 1$ et aller à 1
- ▷ **Expansion** si $(f^r < f^0)$
  - ▷ Construire $y^e = y^c + \delta^e(y^c - y^n)$
  - ▷ Evaluer $f^e = f(y^e)$
  - ▷ Si $(f^e \leq f^r)$
    - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^e\}$
  - ▷ Sinon
    - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^r\}$
  - ▷ $k \leftarrow k + 1$ et aller à 1
- ▷ **Contraction** si $(f^r \geq f^{n+1})$
  - ▷ Si $(f^r < f^n)$: contraction externe
    - ▷ Construire $y^{oc} = y^c + \delta^{oc}(y^c - y^n)$
    - ▷ Evaluer $f^{oc} = f(y^{oc})$
    - ▷ Si $(f^{oc} \leq f^r)$
      - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^{oc}\}$
      - ▷ $k \leftarrow k + 1$, aller à 1
    - ▷ Sinon
      - ▷ Aller à **Rétrécissement**
  - ▷ Si $(f^r \geq f^n)$: contraction interne
    - ▷ Construire $y^{ic} = y^c + \delta^{ic}(y^c - y^n)$
    - ▷ Evaluer $f^{ic} = f(y^{ic})$
    - ▷ Si $(f^{ic} < f^r)$
      - ▷ $Y_{k+1} \leftarrow \{y_k^0, y_k^1, \ldots, y_k^{n+1}, y^{ic}\}$
      - ▷ $k \leftarrow k + 1$, aller à 1
- ▷ **Rétrécissement** si $(f^r \geq f^{n-1})$
  - ▷ Construire les n points $y^{s_i}, i = 1, 2, \ldots, n$
  - ▷ Evaluer f aux n points
  - ▷ $Y_{k+1} \leftarrow \{y^{s_1}, y^{s_2}, \ldots, y^{s_n}\}$
  - ▷ $k \leftarrow k + 1$ et aller à 1

$\mathbb{R}^{(p+1)\times(q+1)}$,

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_q(y^p) \end{bmatrix},$$

cela revient à résoudre $M(\phi, Y)\alpha = f(Y)$, où $f(Y) = (f(y^0), f(y^1), \ldots, f(y^p))^\top$.

Si $p > q$, on dispose de plus de points que nécessaire. On va donc construire le modèle en résolvant

$$\min_{\alpha \in \mathbb{R}^{q+1}} \|M(\phi, Y)\alpha - f(Y)\|^2.$$

Si $p = q$, alors la solution est unique. Si $p < q$, on ne dispose pas de suffisamment de points pour construire le modèle, et il y a une infinité de solutions. C'est la cas le plus fréquent en optimisation de boîtes noires, vu qu'on ne dispose que d'un budget limité d'évaluations. Dans ce cas, on choisis la solution qui minimise la norme de Frobenius de la matrice Hessienne, *i.e.* qui minimise la courbure du modèle.

Etant donné un modèle, on voudrait juger de sa qualité. En supposant une fonction $f \in \mathcal{C}^1$ et ∇f Lipschitz continue, un modèle m_f est appelé complètement linéaire (CL) sur $B(y; \Delta)$ si

$$\begin{cases} |f(x) - m_f(x)| \leq \kappa_f \Delta^2 \\ |\nabla f(x) - \nabla m_f(x)| \leq \kappa_g \Delta \end{cases},$$

pour tout $x \in B(y; \Delta)$ et pour des constants κ_f, $kappa_g$; Δ est appelé paramètre de précision du modèle. D'un manière similaire, on peut définir un modèle complètement quadratique (CQ). Soit $f \in \mathcal{C}^2$ et $\nabla^2 f$ Lipschitz continue, un modèle m_f est complètement quadratique si

$$\begin{cases} |f(x) - m_f(x)| \leq \kappa_f \Delta^3 \\ |\nabla f(x) - \nabla m_f(x)| \leq \kappa_g \Delta^2 \\ |\nabla^2 f(x) - \nabla^2 m_f(x)| \leq \kappa_h \Delta^3 \end{cases},$$

pour tout $x \in B(y; \Delta)$ et pour des constants κ_f, $kappa_g$ et κ_h.

Un ensemble de modèles $\mathcal{M} = \{m : \mathbb{R}^n \mapsto \mathbb{R}, m \in \mathcal{C}^2\}$ est une classe complètement linéaire (resp. quadratique) s'il existe un modèle complètement linéaire (resp. quadratique) dans $\mathcal{M}$ et s'il existe un procédure capable de soit déterminer si un modèle est complètement linéaire (resp. quadratique) sur $B(x; \Delta)$, soit trouver un modèle complètement linéaire (resp. quadratique) sur $B(x; \Delta)$.

Vu qu'un modèle est généralement plus simple, les limitations dues au temps d'évaluation de la boîte noire ne s'appliquent pas aux modèles. Les auteurs de [10] concluent que si les modèles sont de qualité suffisantes, alors il est possible d'adapter des méthodes d'optimisation lisse à l'optimisation sans dérivées. Dans la suite de cette section, on présente une généralisation de la méthode du gradient basée sur des modèles ainsi qu'une méthode de région de confiance.

3.2.1 Descente basée sur des modèles

La méthode du gradient [49, Chapitre 1] est une méthode d'optimisation bien connue. Celle-ci consiste à réaliser une recherche linéaire dans la direction de plus forte descente, opposée au gradient, à chaque itération. Toute la méthode se base sur le calcul du gradient à chaque itération pour trouver la direction de plus forte descente.

Grâce à l'utilisation de modèles, on peut généraliser cette méthode en optimisation sans dérivées. L'idée de base est la suivante. A chaque itération, on construit un modèle local dérivable et on effectue une recherche linéaire dans la direction opposé au gradient du modèle. La qualité du modèle est assez importante pour l'optimisation, c'est pourquoi on vérifie la qualité du modèle avant d'effectuer la recherche linéaire.

On introduit les notations suivantes. $x_0 \in \mathbb{R}^n$ représente le point de départ et $x_k \in \mathbb{R}^n$ l'itéré courant, m_f est un modèle complètement linéaire de précision $\Delta_k > 0$. Un paramètre $\mu_0 > 0$ vérifie la précision de l'évaluation du modèle, $\eta \in]0;1[$ le paramètre d'Armijo TODO : citer , $\epsilon_d \in]0;1[$ angle minimum de descente et $\epsilon \geq 0$ une tolérance d'arrêt. L'algorithme de descente basée sur des modèles est décrit à l'algorithme 3.

Algorithme 3 Descente basée sur des modèles

0. Initialisation
- ▷ Choisir x_0, Δ_0, μ_0, ϵ_d et ϵ
- ▷ $k \leftarrow 0$.

1. Construction du modèle
- ▷ Construire m_f^k à partir de Δ_k et d'un nombre de points

2. Vérification du modèle
- ▷ Si ($\Delta_k < \epsilon$ et $\|\nabla m_f^k(x_k)\| < \epsilon$) : Stop
- ▷ Si ($\Delta_k > \mu \|\nabla m_f^k(x_k)\|$) :
  - ▷ Modèle pas précis
  - ▷ $\Delta_{k+1} \leftarrow \Delta_k/2, \mu_{k+1} \leftarrow \mu_k, x_{k+1} \leftarrow x_k$
  - ▷ $k \leftarrow k+1$, aller à l'étape 1
- ▷ Si ($\Delta_k \leq \mu \|\nabla m_f^k(x_k)\|$) :
  - ▷ Modèle précis
  - ▷ Aller à l'étape 3

3. Recherche linéaire
- ▷ Choisir d_k tel que $\left(\frac{d_k}{\|d_k\|}\right)^\top \left(\frac{\nabla m_f^k(x_k)}{\|\nabla m_f^k(x_k)\|}\right) < -\epsilon_d$
- ▷ Chercher t_k tel que $f(x_k + t_k d_k) < f(x_k) + \eta t_k d_k^\top \nabla m_f^k(x_k)$

4. Mise à jour
- ▷ Si t_k est trouvé :
  - ▷ $x_{k+1} \leftarrow y$ où $f(y) \leq f(x_k + t_k d_k)$
  - ▷ $\mu_{k+1} \leftarrow \mu_k$
- ▷ Sinon :
  - ▷ $x_{k+1} \leftarrow x_k$
  - ▷ $\mu_{k+1} \leftarrow \mu_k/2$
- ▷ $\Delta_{k+1} \leftarrow \Delta_k$
- ▷ $k \leftarrow k+1$, aller à l'étape 1

3.2.2 Méthodes de région de confiance

Les méthodes de région de confiance exploite un modèle simple, en général lisse et facile à évaluer, et suppose qu'il a un comportement similaire à l'objectif dans un voisinage, la région de confiance, de la solution courante.

Un modèle linéaire peut être utilisé car il n'a besoin que de $\mathcal{O}(n)$ points pour être construits mais il ne donne aucune information quant à la courbure du problème. Un modèle quadratique est donc en général privilégié, de la forme :

$$m_f(x) = f(x_k) + g_k^\top (x - x_k) + (x - x_k)^\top H_k (x - x_k)^\top,$$

où $g_k \in \mathbb{R}^n$ et $H_k \in \mathbb{R}^{n \times n}$ symétrique. Ces paramètres correspondent à gradient et à la Hessienne du modèle en$x = 0$ et sont estimés en imposant que le modèle interpole un certain ensemble de points, $m_f(x^i) = f(x^i)$, $i = 0, 2, \ldots, p$.

Le modèle est supposé fidèle dans un voisinage du point x_k, en général ce voisinage est pris comme la boule de rayon Δ_k,

$$B(x_k, \Delta_k) = \{x \in \mathbb{R}^n : \|x - x_k\| \leq \Delta_k\}.$$

A chaque itération, on cherche à minimiser ce modèle dans la boule de rayon de Δ_k. Bien entendu, le modèle n'est pas toujours très fidèle à l'objectif. C'est pourquoi on calcule un rapport r qui compare

la réduction prévue par le modèle et celle observée par l'objectif. Pour un point t, ce rapport est défini comme

$$r(t) = \frac{f(x_k) - f(t)}{m_f(x_k) - m_f(t)}.$$

Si ce rapport n'est pas suffisant, le modèle est considéré comme pas assez fidèle et soit la taille de la région de confiance est réduite, soit on applique une procédure sensé amélioré la qualité du modèle. L'algorithme complet est décrit à l'algorithme 4.

Algorithme 4 Algorithme de région de confiance sans dérivées

0. Initialisation
- ▷ Choisir une classe de modèle
- ▷ Choisir une méthode d'amélioration du modèle
- ▷ Données : x_0, Δ_{max}, $\Delta_0 \in]0; \Delta_{max}$
- ▷ Initialisation le modèle m_f
- ▷ $k \leftarrow 0$

1. Test critique
- ▷ Si $\|g_k\| \leq \epsilon$
  - ▷ Appliquer la méthode d'amélioration du modèle
  - ▷ Si le modèle n'est pas assez bon ou la région de confiance trop grande
    - ▷ Construction d'un nouveau modèle
  - ▷ Sinon
    - ▷ Vérifier le critère d'arrêt → STOP

2. Optimisation du sous-problème
- ▷ Trouver $t \in \operatorname{argmin}_{x \in B(x_k;\Delta_k)} m_f(x)$
- ▷ Evaluer $f(t)$ et calculer le rapport $r(t)$

3. Acceptation du candidat
- ▷ Si $r(t) \geq \eta_1$
  - ▷ $x_{k+1} \leftarrow t$, mise à jour du modèle avec le point t
- ▷ Sinon
  - ▷ $x_{k+1} \leftarrow x_k$

4. Amélioration du modèle
- ▷ Si $r(t) < \eta_1$
  - ▷ Appliquer la procédure d'amélioration du modèle

5. Mise à jour de la région de confiance

$$\Delta_{k+1} \in \begin{cases} [\Delta_k; \min\{\gamma_{inc}\Delta_k; \Delta_{max}\}] & \text{si } r(t) \geq \eta_1 \\ \{\gamma_{dec}\Delta_k\} & \text{si } r(t) < \eta_1 \text{ et } m_f \text{ est CL} \\ \{\Delta_k\} & \text{si } r(t) < \eta_1 \text{ et } m_f \text{ n'est pas CL} \end{cases}$$

- ▷ $k \leftarrow k + 1$

3.2.3 Optimisation Bayesienne

Les méthodes d'optimisation Bayesienne se basent elles aussi sur des modèles. Toutefois, il s'agit plutôt de modèle probabilistes.

En optimisation Bayesienne, la fonction objectif est considérée comme aléatoire, puisque celle-ci n'est pas connue. On suppose donc que la fonction objectif suit une distribution *a priori*, qui représente ce que l'on suppose de la fonction objectif. Une fois que la boîte noire est évaluée, la distribution de la fonction objectif est mise à jour pour former une distribution *a posteriori*. Les distributions *a priori* et *a posteriori* sont liées via la règle de Bayes. On suppose une variable aléatoire F avec une distribution *a priori* $p(F)$. Celle-ci représente notre croyance quant aux possibles valeurs que pourrait prendre F avant que celle-ci ait été observée. Ensuite, on suppose que l'on a accès à des données D et un modèle

de vraisemblance $p(D|F)$. On peut donc construire une distribution *a posteriori* $p(F|D)$,

$$p(F|D) \propto p(D|F)p(F).$$

La distribution *a posteriori* est ensuite utilisée pour construire une fonction d'acquisition u qui déterminera de nouveaux points à évaluer. Ces nouveaux points seront utilisés pour mettre à jour les distributions à l'itération suivante. Une description de haut niveau de cette méthode est présentée à l'algorithme 5.

Il existe plusieurs manières de construire les distributions *a priori* et *a posteriori* ; parmi les plus connues on note les processus Gaussiens ou les processus de Wiener . Plusieurs modèles sont présentés dans [56, section 2 et 3] et dans [18, section 2]. La nature de la fonction d'acquisition est également importante. En effet, les nouveaux points à évaluer sont déterminés par celle-ci, soit car la valeur de l'objectif est intéressante soit car l'incertitude quant à celle-ci est importante. Cela permet une balance naturelle entre l'exploration de l'espace de recherche et l'intensification de la recherche à un endroit plus précis. Les exemples les plus connus consistent à maximiser la probabilité d'amélioration, une espérance d'amélioration de la fonction ou une borne de confiance [45]. Une revue de possibles fonctions d'acquisition est présentée dans [56, section 4] et [18, section 2.3].

Algorithme 5 Optimisation Bayesienne

0. Initialisation
- ▷ Ensemble de données $D_0 = \emptyset$
- ▷ Fonction d'acquisition u
- ▷ $k \leftarrow 0$

1. Optimisation
- ▷ Trouver $x_k \in \text{argmax}_x\, u(x|D_k)$ à l'aide de la distribution *a priori*
- ▷ Evaluer l'objectif $y_k = f(x_k)$

2. Mise à jour
- ▷ $D_{k+1} = \{D_k, (x_k, y_k)\}$
- ▷ Mettre à jour

TODO : Exemple d'utilisation et de performance des méthodes

3.3 Méthodes de recherche directe

Les *méthodes de recherche directe* sont un ensemble d'algorithmes d'optimisation qui ont la structure suivante. Au début d'une itération, l'algorithme connaît une *solution courante*. Il s'agit de la meilleure solution connue par la méthode. Un ensemble de points d'essais est évalué. Ensuite, des actions sont prises en fonction des évaluations de ces points ; si un de ces points améliore la solution courante, celle-ci est mise à jour, sinon un paramètre de taille de pas est réduit et un nouvel ensemble de points d'essais est généré. Il faut noter qu'aucun modèle de la fonction n'est construit pour générer l'ensemble de points à évaluer, ni aucune approximation des dérivées. Dans cette section, on présente plusieurs méthodes de recherche directe allant des plus simples, comme la recherche par coordonnées, jusqu'à des méthodes plus évoluées comme la recherche par treillis adaptatif.

3.3.1 Recherche par coordonnées

Une prmière méthode de recherche directe assez simple est la recherche par coordonnées [29]. On note x_k la solution courante à l'itération k. Chaque itération est composée d'une sonde, il s'agit d'évaluer les points $x_k \pm \delta_k e_i$, où e_i est le vecteur de $\mathbb{R}^n$ composé uniquement de 0 et d'un 1 à la i^e composante. Si un point améliorant la solution courante est trouvé, celui-ci devient la nouvelle solution courante et l'itération est considérée comme un succès ; sinon le paramètre de pas δ_k est diminué et une nouvelle sonde est effectuée. Certaines variantes de cet algorithme existent. Par exemple, le paramètre de pas pourrait être augmenté lors des succès ou la sonde peut être interrompue dès qu'un point

améliorant la solution est trouvée. Cette dernière variante est appelée stratégie opportuniste. Lorsque cette stratégie est appliquée, l'ordre d'évaluations des points de l'ensemble de sonde a une certaine importance. La recherche par coordonnées est décrite à l'algorithme 6.

Algorithme 6 Recherche par coordonnées

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\delta^0 \in (0; \infty)$	Taille de pas initiale
$\epsilon_{stop} \in [0; \infty)$	Tolérance d'arrêt
$k \leftarrow 0$	Compteur d'itération

1. Sonde

▷ Si $f(t) < f(x^k)$ pour un certain $t \in P^k := \left\{x^k \pm \delta^k e_i : i = 1, 2, \ldots, n\right\}$

  ▷ $x^{k+1} \leftarrow t$ et $\delta^{k+1} \leftarrow \delta^k$

▷ Sinon

  ▷ $x^{k+1} \leftarrow x^k$ et $\delta^{k+1} \leftarrow \frac{\delta^k}{2}$

2. Terminaison

▷ Si $\delta^{k+1} \geq \epsilon_{stop}$

  ▷ Incrémenter $k \leftarrow k + 1$ et aller à 1

▷ Sinon

  ▷ Stop

Pour la recherche par coordonnées, il existe un résultat de convergence assez faible. Celui-ci est décrit au théorème 3.2.

Théorème 3.2 (Convergence de la recherche par coordonnées) *Soient $f : \mathbb{R}^n \mapsto \mathbb{R}$ une fonction $\mathcal{C}^0$ avec des ensembles de niveau bornés et $\{x^k\}$ la suite produite par la méthode de recherche par coordonnées avec $\epsilon_{stop} = 0$. Soit $\hat{x}$ la point d'accumulation des itérations échec de $\{x^k\}$.*

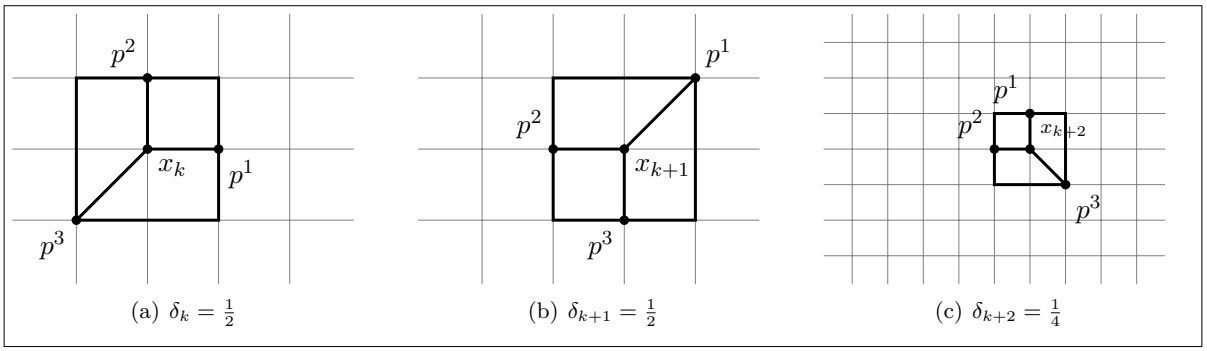
Alors, pour tout $d \in \{\pm e_i : i = 1, 2, \ldots, n\}$, soit $f'(\hat{x}; d) \geq 0$, soit $f'(\hat{x}; d)$ n'existe pas. De plus, si $f \in \mathcal{C}^1$, alors $\hat{x}$ est un point critique de f, i.e. $\nabla f(\hat{x}) = 0$.

Ce résultat est intéressant mais relativement faible car il requiert une fonction de classe $\mathcal{C}^0$. Par exemple, on considère la fonction convexe $f : \mathbb{R}^2 \mapsto \mathbb{R}, f(x) = \|x\|_\infty = \max(|x_1|, |x_2|)$, et le point $x^0 = [1, 1]^\top$. Quelle que soit la valeur de δ, $x^0 \pm e_i$ (pour $i = 1, 2$) n'améliore pas la solution courante $f(x^0) = 1$. En effet, les directions e_i $(i = 1, 2)$ sont des directions de montée tandis que les directions $-e_i$ sont parallèles aux ensemble de niveau. La recherche par coordonnées ne pourra donc trouver le minimum $[0, 0]^\top$ de la fonction.

La suite de cette section présente les méthodes de recherche par motif généralisée et de recherche directe par treillis adaptatif. Ces méthodes ont une structure similaire à la recherche par coordonnées mais possèdent des résultats de convergence plus intéressants.

3.3.2 Recherche par motif généralisée

La *recherche par motif généralisée* (Gps) [60] est une amélioration de la recherche par coordonnées. Concrètement, elle propose d'autres directions de recherche que celles alignées sur les axes, ainsi que la possibilité pour ces directions de changer en fonction des itérations. En plus de l'étape de sonde, une recherche locale, la recherche par motif généralisée propose une étape de recherche plus globale appelée simplement étape de recherche. Cette recherche consiste à évaluer un nombre fini de points dans l'espace de recherche. Si l'étape de recherche génère un point améliorant la solution courante, il s'agit d'un succès, l'étape de sonde peut être passé. L'étape de sonde de Gps est assez similaire à l'étape de sonde de la recherche par coordonnées. Un ensemble de points autour de la solution courante est généré et évalué en espérant trouver une meilleure solution. La différence porte sur la façon de générer cet ensemble de sonde. Pour assurer des résultats de convergence intéressant, il faut contrôler les étapes de sonde et de recherche. Cela est fait au travers d'un treillis, défini à la définition 3.1.



(a) $\delta_k = \frac{1}{2}$ (b) $\delta_{k+1} = \frac{1}{2}$ (c) $\delta_{k+2} = \frac{1}{4}$

FIGURE 3.1 – Représentation du treillis et cadre de sonde de GPS

Définition 3.1 (Treillis ([10, chapitre 7], notre traduction) *Soit $G \in \mathbb{R}^{n\times n}$ une matrice inversible et $Z \in \mathbb{Z}^{n\times p}$ telle que les colonnes de Z forment un ensemble générateur positif de $\mathbb{R}^n$. On définit $D = GZ$. Le* treillis *généré par D centré sur la solution courante $x^k \in \mathbb{R}^n$ de taille de maille $\delta^k > 0$ est défini par*

$$M^k := \left\{x^k + \delta^k Dy : y \in \mathbb{N}^p\right\} \subset \mathbb{R}^n.$$

Puisque les colonnes de la matrice $Z \in \mathbb{Z}^{n\times p}$ forment un ensemble générateur positif et que la matrice G est inversible, les colonnes de la matrice D forment également un ensemble générateur positif. On appelle D une matrice générateur positif et on note $\mathbb{D}$ l'ensemble de directions composé des colonnes de D. On note que tous les points évalués appartiennent au treillis. Il faut noter que le treillis est un concept mais ne doit pas être construit en tant que tel au cours de l'exécution de l'algorithme. Une illustration de trois sondes successives est présentée à la figure 3.1. La solution courante et les points de sonde sont représentés par le symbol "•". Le cadre noir retient l'ensemble des points du treillis qui peuvent être évalués lors de la sonde. Lors de la première sonde sur la figure 3.1(a), le point p^1 améliore la solution. C'est pourquoi le cadre de s'est déplacé sur la figure 3.1(b). Lors de la deuxième sonde, aucun point améliorant la solution n'est trouvé et le treillis est raffiné, comme cela est représenté sur la figure 3.1(c). La méthode de recherche par motif généralisée est décrite à l'algorithme 7.

Pour analyser la convergence de cette méthode, on s'intéresse à des situations où le nombre d'itérations tend vers l'infini. La convergence de cette méthode repose sur les arguments suivants. On suppose que l'on cherche à minimiser une fonction dont l'ensemble de niveau $\{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ forme un ensemble compact. Alors, le nombre d'itérations où une meilleure solution courante est trouvée, un succès, est fini. En effet, puisque lors d'un succès, la fonction objectif est strictement réduite et que le treillis est une structure discrète, il n'y a qu'un nombre fini d'itérations formant un succès. Lorsque le nombre d"itérations tend vers l'infini, on peut en conclure que le nombre d'itérations formant des échecs tend également vers l'infini. Puisque la taille du maillage δ_k est réduite à chaque itération infructueuse, on peut déduire une limite quant à cette taille de maillage du treillis,

$$\lim_{k\to\infty} \inf \delta^k = 0. \tag{3.1}$$

De plus, il existe une sous-suite d'itérations infructueuses $\{k_i\}$ et un point x^* tels que

$$\lim_{i\to\infty} \delta^{k_i} = 0, \quad \lim_{i\to\infty} x^{k_i} = x^*.$$

Ceci vient du fait qu'il y a une infinité d'itérations où l'étape de sonde échoue et que la taille du maillage est réduite uniquement lors de ces itérations. On note K_e^1 la sous-suite d'itérations échecs. Si la limite (3.1) est satisfaite, il existe une sous-suite $K_e^2 \subset K_e^1$ tel que $\delta_k \to 0$, pour tout $k \in K_e^2$. La suite $\{x_k\}_{K_e^2}$ est donc bornée et contient une sous-suite convergente $\{x_k\}_{K_e^3}$. On note $x^* = \lim_{k\in K_e^3} x_k$.

Cela implique que l'algorithme raffine le maillage du treillis dans le voisinage de la solution courante. En supposant que la fonction objectif est localement Lipschiptz autour du point x^*, on peut en conclure un résultat quant aux dérivées directionnelles de Clarke [20],

$$f^\circ(x^*; d) \geq 0, \quad \forall d \in \mathbb{D}^*, \tag{3.2}$$

Algorithme 7 Recherche par motif généralisée

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\delta^0 \in (0; \infty)$	Paramètre de treillis intial
$D = GZ$	Matrice générateur positif
$\tau \in (0; 1), rationnel$	Paramètre d'ajustement du treillis
$\epsilon_{stop} \in [0; \infty)$	Tolérance d'arrêt
$k \leftarrow 0$	Compteur d'itération

1. Recherche

▷ Si $f(t) < f(x^k)$ pour un certain t d'un sous-ensemble fini du treillis M^k

  ▷ $x^{k+1} \leftarrow x^k$, $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$, et aller à 3

▷ Sinon

  ▷ Aller à 2

2. Sonde

▷ Choisir un ensemble générateur positif $\mathbb{D}^k \subseteq \mathbb{D}$

▷ Si $f(t) < f(x^k)$ pour un certain $t \in P^k := \left\{x^k \pm \delta^k d : d \in \mathbb{D}^k\right\}$

  ▷ Alors $x^{k+1} \leftarrow t$ et $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$

▷ Sinon x^k est un minimum local du treillis

  ▷ $x^{k+1} \leftarrow x^k$ et $\delta^{k+1} \leftarrow \tau\delta^k$

3. Terminaison

▷ Si $\delta^{k+1} \geq \epsilon_{stop}$

  ▷ incrémenter $k \leftarrow k + 1$ et aller à 1

▷ Sinon

  ▷ Stop

où $\mathbb{D}^*$ est un ensemble générateur positif de D et

$$f^\circ(x;d) = \limsup_{y \to x, t \downarrow 0} \frac{f(y+td) - f(y)}{t}.$$

En imposant que la fonction objectif est strictement dérivable au point x^*, on obtient

$$\nabla f(x^*) = 0.$$

Ce résultat de convergence est plus général que celui de la recherche par coordonnées mais souffre d'un défaut majeur. La matrice générateur positif D est de dimension finie. Il existe donc un ensemble fini de générateur positif et donc de direction explorée autour de la solution x^*.

Une étude plus complète de la convergence de Gps peut être trouvée dans [10, 22, 7, chapitre 7].

3.3.3 Recherche directe sur treillis adaptatif

Bien que la recherche par motif généralisée généralise la recherche par coordonnées en sondant dans une plus grande variété de directions, les résultats de convergence ne sont pas encore à la hauteur de nos espérances. En effet, l'initialisation de Gps requiert une matrice générateur positif. Toutes les directions de sonde vont être sélectionnées parmi les colonnes de cette matrice, ce qui implique un nombre fini de directions de sonde dans Gps.

Pour généraliser la recherche par motif généralisée, la recherche directe par treillis adaptatif (Mads) [8] propose un moyen d'avoir une infinité de directions de sonde possibles. Cela permet de contrecarrer les limites de convergence de Gps. Un autre avantage de Mads est sa gestion des contraintes notamment via la barrière extrême.

Contrairement à Gps, la recherche directe par treillis adaptatif possède un paramètre pour la taille du treillis δ^k et un paramètre pour la taille du cadre de sonde Δ^k. Celui-ci n'était pas nécessaire dans Gps car Gps est un cas particulier de Mads où ces deux paramètres ont la même valeur.

Mads introduit donc les notions de cadre de sonde et de taille du cadre de sonde reprises dans la définition 3.2.

Définition 3.2 (Cadre de sonde ([10, section 8.1], notre traduction) *Soit $G \in \mathbb{R}^{n \times n}$ une matrice inversible et $Z \in \mathbb{Z}^{n \times p}$ telle que les colonnes de Z forment un ensemble générateur positif de $\mathbb{R}^n$. On définit $D = GZ$. On choisit un paramètre de taille de treillis $\delta^k > 0$ et on définit Δ^k tel que $\delta^k \leq \Delta^k$. Le* cadre *de taille Δ^k généré par D centré sur la solution courante $x^k \in \mathbb{R}^n$ est défini par*

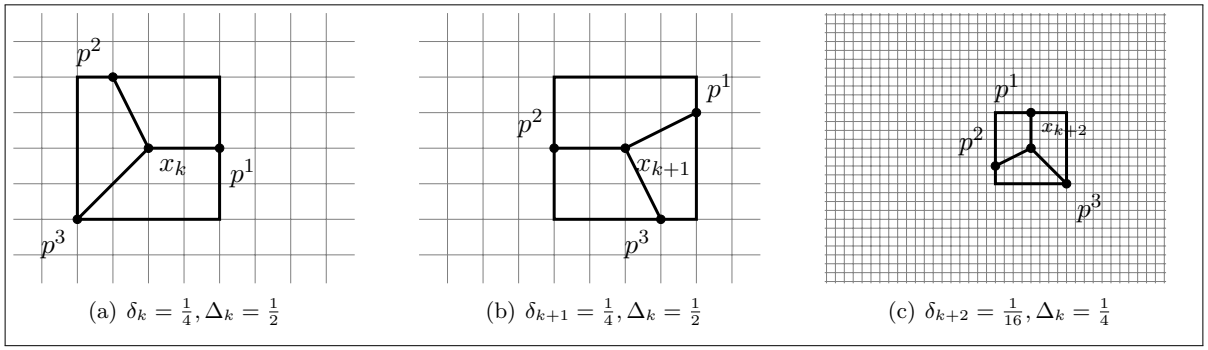
$$F^k := \left\{ x \in M^k : \|x - x^k\|_\infty \leq \Delta^k b \right\}$$

avec $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$ et où Δ^k est appelé paramètre de taille du cadre.

On note que l'ensemble de sonde de Gps est inclus dans le cadre de sonde de Mads, qui est lui-même est sous-ensemble du treillis pour un treillis et une solution courante donnés. En réduisant plus rapidement le paramètre de taille du treillis que celui de taille du cadre, un plus grand ensemble de directions de sonde peuvent être sélectionnées, de manière à créer un ensemble de directions de sonde dense dans la sphère unitaire. Les paramètres de taille du treillis et de sonde doivent respecter $0 < \delta^k \leq \Delta^k$ à chaque itération et

$$\lim_{k \in K} \delta_k = 0 \iff \lim_{k \in K} \Delta_k = 0, \text{pour tout sous-ensemble d'indices } K.$$

Pour cela, la stratégie $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$ est proposée, bien que d'autres stratégies peuvent être possible. Pour illustrer cela, on présente une figure similaire à la figure 3.1, mais adaptée à la situation de Mads. Celle-ci est présentée à la figure 3.2. On peut y voir qu'il y a plus de directions de sonde possibles. De plus, lorsque le treillis est raffiné à la figure 3.2(c), on passe de $25 = (4+1)^2$ à $81 = (8+1)^2$ points du treillis qui se trouvent dans le cadre de sonde. Ceci montre la densité des directions de sonde lorsque Δ_k tend vers 0. La méthode de recherche directe par treillis adaptatif est présentée à l'algorithme 8.



(a) $\delta_k = \frac{1}{4}, \Delta_k = \frac{1}{2}$ (b) $\delta_{k+1} = \frac{1}{4}, \Delta_k = \frac{1}{2}$ (c) $\delta_{k+2} = \frac{1}{16}, \Delta_k = \frac{1}{4}$

FIGURE 3.2 – Représentation du treillis et cadre de sonde de MADS

Algorithme 8 Recherche directe par treillis adaptatif

Données : objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ et un point de départ $x^0 \in \mathbb{R}^n$

0. Initialisation

$\Delta^0 \in (0; \infty)$	Paramètre de cadre initial
$D = GZ$	Matrice générateur positif
$\tau \in (1; \infty), rationnel$	Paramètre d'ajustement du treillis
$\omega^+ \geq 0, \omega^- \leq -1$	Paramètres d'ajustement du treillis
$\epsilon_{stop} \in [0; \infty)$	Tolérance d'arrêt
$k \leftarrow 0$	Compteur d'itération

1. Mise à jour du paramètre

▷ Définir le paramètre de taille du treillis $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$

2. Recherche

▷ Si $f(t) < f(x^k)$ pour un certain t d'un sous-ensemble fini S^k du treillis M^k

  ▷ $x^{k+1} \leftarrow x^k$, $\Delta^{k+1} \leftarrow \tau^{\omega^+}\Delta^k$, et aller à 4

▷ Sinon

  ▷ Aller à 3

3. Sonde

▷ Choisir un ensemble générateur positif $\mathbb{D}^k_\Delta$ tel que $P^k = \{x^k + \delta^k d : d \in \mathbb{D}^k_\Delta\}$ soit un sous-ensemble du cadre F^k de taille Δ

▷ Si $f(t) < f(x^k)$ pour un certain $t \in P^k$

  ▷ Alors $x^{k+1} \leftarrow t$ et $\Delta^{k+1} \leftarrow \tau^{\omega^+}\Delta^k$

▷ Sinon x^k est un minimum local du treillis

  ▷ $x^{k+1} \leftarrow x^k$ et $\Delta^{k+1} \leftarrow \tau^{\omega^-}\Delta^k$

4. Terminaison

▷ Si $\Delta^{k+1} \geq \epsilon_{stop}$

  ▷ incrémenter $k \leftarrow k + 1$ et aller à 1

▷ Sinon

  ▷ Stop

L'analyse de convergence de MADS est similaire à celle de GPS. La différence la plus importante repose sur le fait qu'il y a une infinité de directions de sonde possibles. De plus, la différentiation du cadre de sonde et du treillis permet d'avoir un ensemble de directions de sonde plus riche que GPS, et donc un résultat de convergence plus fort. En effet, puisque la paramètre de taille du treillis doit toujours être plus petit que le paramètre de taille du cadre de sonde, plus de directions peuvent être utilisées pour créer un ensemble générateur positif. A la limite, cet ensemble de directions de sonde possibles devient dense dans la sphère unité. L'analyse de convergence de GPS tient également pour MADS. Puisqu'il y a un plus grand ensemble de directions de sonde possible, le résultat de convergence de GPS (3.2) peut être étendu pour MADS en

$$f^\circ(x^*; d) \geq 0, \quad \forall d \in \mathbb{R}^n.$$

Une analyse de convergence complète de MADS est reprise dans [10, 22, chapitre 7] ou [8]. La qualité de MADS dépend de la manière de générer les ensembles générateur positif aux étapes de sonde. Différentes implémentations sont présentées dans [8, 2, 40, 62].

Il est important de noter que l'analyse de convergence de GPS et de MADS se base essentiellement sur l'étape de sonde. L'étape de recherche doit uniquement satisfaire les deux conditions suivantes :

1. Evaluer un nombre fini de points,
2. Tous les points évalués doivent appartenir au treillis.

Le cadre algorithmique de MADS laisse donc de grandes libertés permettant d'intégrer des stratégies de recherche diverses et variées comme une recherche basée sur des modèles ou sur une heuristique, tant que celles-ci satisfont ces deux conditions.

Par exemple, il est possible de lié la méthode de Nelder-Mead et un algorithme MADS. En effet, l'étape de recherche de MADS peut être basée sur la méthode de Nelder-Mead. Ceci est présenté dans [12]. Une autre possibilité, présentée dans [21], est d'appliquer une étape de recherche basée sur des modèles quadratiques.

3.4 Optimisation sans dérivées en grande dimension

Dans cette section, nous intéressons aux méthodes d'optimisation sans dérivées qui s'appliquent plus spécifiquement aux problèmes en grande dimension. Une approche assez simple serait d'utiliser la parallélisation. Cela permet d'effectuer plusieurs évaluations de la boîte noire simultanément et limiter le temps d'exécution des algorithmes. En pratique, il existe d'autres stratégies adaptées à partir des méthodes présentées aux sections précédentes qui appliquent des approches plus subtiles.

Dans ce travail, nous intéressons plus particulièrement aux stratégies de réduction de dimension pour pouvoir résoudre des problèmes de grande de taille, sans exploiter le parallélisme. Par exemple, une adaptation de la méthode de descente basée sur des modèles en grande dimension est présentée dans [63]. Dans ce document, on s'intéresse plus particulièrement à des méthodes possédant des résultats de convergence intéressants. Nous avons donc décidé de s'attarder sur des stratégies de réduction de dimension pour les algorithmes d'optimisation Bayesienne et MADS, présentés aux sections 3.2.3 et 3.3.3 respectivement. On présente également une version de l'algorithme de Nelder-Mead adapté pour des problèmes de grande dimension.

3.4.1 Algorithme de Nelder-Mead en grande dimension

Dans [43], Mehta explique que l'algorithme de Nelder-Mead avec ses paramètres mis à leur valeur par défaut a de mauvaises performances sur les problèmes en grande dimension. Les auteurs de [30] expliquent que les opérations d'expansion et de contraction ont une certaine propriété de descente mais que l'efficacité de celle-ci diminue lorsque la taille du problème augmente. Dans ce cas, l'algorithme est alors dominé par l'opération de réflexion. Plusieurs stratégies ont alors été proposées pour fixer les paramètres de l'algorithme. Celles-ci se basent principalement sur des paramètres qui s'adaptent à la taille du problème.

Les auteurs de [27] ont appliqué une algorithme génétique pour faire évoluer l'algorithme de Nelder-Mead tandis que dans [39], on exploite une analyse de sensibilité pour fixer ces paramètres. Dans [43],

Mehta présente une manière de fixer ces paramètres en se basant sur les points de Chebyshev. Une approche pour améliorer les performances de l'algorithme est de perturber le centroïde aléatoirement ce qui améliore la convergence, comme présenté dans [28].

3.4.2 Réduction de dimension en optimisation Bayesienne

En plus des versions exploitant le parallélisme, il existe également des méthodes d'optimisation Bayesienne appliquant des stratégies de réduction de dimension.

Celles-ci supposent que l'objectif a une dimension effective petite comparée à sa dimension. Concrètement, une fonction objectif $f : \mathbb{R}^n \mapsto \mathbb{R}$ possède une dimension effective $n_e \leq n$ s'il existe un sous-espace linéaire T de dimension n_e tel pour tous vecteurs $x \in T$ et $x_\top \in T_\top$,

$$f(x + x_\top) = f(x).$$

La dimension effective n_e est la plus petite dimension satisfaisant cette égalité. On parle également de fonctions avec un sous-espace actif T.

Au lieu d'essayer de résoudre le problème orignal

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s. t.} \quad & x \in \mathcal{X} \end{aligned}$$

où $\mathcal{X}$ désigne des bornes sur les variables, on cherche à résoudre le problème

$$\begin{aligned} \min_{y \in \mathbb{R}^{n_e}} \quad & f(Ay) \\ \text{s. t.} \quad & y \in \mathcal{Y} \end{aligned}$$

où $A \in \mathbb{R}^{n \times n_e}$ et $\mathcal{Y}$ représente le nouvel espace réalisable.

La difficulté vient du fait que la matrice A ainsi que la dimension effective ne sont pas connues. La matrice A est souvent choisie comme un matrice aléatoire [47, 19]. Le choix du domaine de petite dimension $\mathcal{Y}$ peut aussi avoir un impact sur les performances de l'algorithme [15].

TODO : Check reference vers BO avec fonction (partiellement) séparable

TODO : Quantile guassian process pour problèmes en grande dimension [45]

3.4.3 Algorithme Stats-Mads

Comme précisé précédemment, le cadre algorithmique de Mads offre une certaine flexibilité quant à l'étape de recherche. En effet, celle-ci n'est pas indispensable et doit respecter certaines conditions pour pouvoir préserver l'analyse de convergence de la méthode.

L'algorithme Stats-Mads, décrit dans [65, 3] est une instance de Mads développée pour résoudre des problèmes d'optimisation sans dérivées de grande dimension. Stats-Mads se base sur l'idée d'identifier les variables les plus importantes et d'alterner entre une optimisation en n dimensions et une optimisation sur un sous-espace.

On note $I = \{1, 2, \ldots, n\}$ l'ensemble des indices des n variables du problème. $J_l \subset I$ désigne un sous-ensemble de variables. Au cours de l'optimisation, lancer une instance de Mads sur J_l signifie que toutes les variables $j \subset I \setminus J_l$, les variables dont l'indice n'appartient pas à J_l, sont fixées à $\hat{x}_j$, la solution courante. La nouvelle instance de Mads est donc une instance en $|J_l|$ dimension.

L'identification de ces variables prépondérantes se fait en appliquant une méthode d'analyse de sensibilité basée sur une analyse de variance. Pour chaque variables, on calcule le rapport de corrélation, décrit à la section 2.3.1, qui indique la sensibilité de la boîte noire par rapport à cette variable. Ensuite, les variables sont classées par ordre croissant de sensibilité et les pn premières sont fixées, où p est un paramètre défini par l'utilisateur qui définit la proportion de variables devant être fixées. Deux autres paramètres doivent être défini ; n^I désigne le nombre maximal d'évaluations de la boîte noire dans l'espace entier et n^J désigne le nombre maximal d'évaluations dans le sous-espace. L'algorithme Stats-Mads est décrite à l'algorithme 9.

Algorithme 9 Algorithme Stats-Mads

0. Initialisation

$\Delta^0 \in (0; \infty)$	Paramètre de cadre initial
$x^0 \in \mathcal{X}$	point initial
n^I	nombre maximum d'évaluations de la boîte noire dans l'espace entier
n^J	nombre maximum d'évaluations de la boîte noire dans le sous-espace

1. Boucle : Pour $l = 0, 1, 2, \ldots,$

- ▷ Lancer une instance de Mads à partir de x^l avec taille de treillis initial Δ^l et budget n^I ;
- ▷ soit $\hat{x}^l$ la meilleure solution trouvée et Δ_{l+1} la taille de treillis final ;
- ▷ Calculer les indices de sensibilité et définir $J_l \subset I$;
- ▷ Lancer Mads sur J_l à partir de $\hat{x}^l$ avec taille de pas initial Δ_l et budget n^J ;
- ▷ soit x_{l+1} la meilleure solution trouvée ;

Les travaux présentés dans [65, 3] montrent que l'algorithme Stats-Mads améliore effectivement les performances Mads sur un ensemble de problèmes d'optimisation non contraints.

Une autre approche pour s'attaquer aux problèmes d'optimisation sans dérivées est d'utiliser le parallélisme. En effet, cela permet par exemple d'effectuer plusieurs évaluations de la boîte noire simultanément. De plus amples informations sur la parallélisme sont disponibles dans [9, 6]. Dans la version parallèle de Mads, l'espace de recherche est divisé de telles manières à ce qu'un processeur n'ait qu'un petit nombre de variables à traiter. Une version de cet algorithme applique une méthode d'analyse de sensibilité pour déterminer l'attribution des variables aux processeurs disponibles ; cet algorithme est décrit dans [5, 4].