POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Opportunisme et traitement des contraintes dans MADS

STÉPHANE JACQUET
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Avril 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Opportunisme et traitement des contraintes dans MADS

présentée par **Stéphane JACQUET**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Sébastien LE DIGABEL, président
Charles AUDET, membre et directeur de recherche
Gilles CAPOROSSI, membre et codirecteur de recherche
Jonathan JALBERT, membre
Jean BIGEON, membre externe

DÉDICACE

À mes professeurs et à mes proches, qui m'ont accompagné avant et pendant ce projet doctoral.

REMERCIEMENTS

Je tiens à remercier mes directeurs, Charles Audet et Gilles Caporossi. Ils m'ont soutenu dans ma volonté de faire un doctorat. Ils ont été très disponibles ; je pouvais taper à la porte de leur bureau quand je voulais le souhaitais. Ils m'ont toujours aidé à avoir identifier des pistes quand j'étais bloqué et à fixer les priorités quand j'étais dans le brouillard.

Je remercie Sébastien Le Digabel, Jean Bigeon et Jonathan Jalbert pour avoir accepté d'être dans mon jury de thèse, ainsi que d'avoir proposé des corrections et suggestions pertinentes.

Un grand merci à Christophe et Vivianne pour les réponses à mes nombreuses questions concernant NOMAD. J'ai une pensée toute particulière pour Nadir et Vilmar, que j'ai suivis de bureau en bureau au GERAD. Les discussions (y compris sur le notamment de football. . .) ont animé nos journées. Ils m'ont beaucoup aidé aussi sur le code quand j'avais l'impression d'avoir tout testé. Je remercie aussi les autres collègues au GERAD : Michael, Christian, Mariana, Mathilde, Sara, Lê, Marylène, Ludovic, Eloïse, Juan, Jesus, Filippo, Jaime, ainsi que tous ceux avec lesquels j'ai partagé de bons moments.

Je remercie beaucoup mes parents. Leur soutien dans les moments difficiles a été primordial. Il y a également mes frères qui m'ont beaucoup partagé avec moi leurs expériences personnelles pour gérer au mieux mon doctorat. Merci aussi à mon cousin Nicolas pour tous les encouragements qu'il m'a apportés au cours de ces années.

Un grand merci à Géraldine pour le réconfort lors des derniers mois du doctorat, et qui a pris le temps de faire la première relecture de ma thèse au complet.

J'ai une pensée pour tous mes amis de Paris (la liste est trop longue ici), mais je tiens à remercier tous mes amis, hors du GERAD, que j'ai ici : Victor, Sarah, Maud, Pierre-Alexandre, Florian, Alexandre et Dan.

Enfin, si j'ai toujours eu du plaisir à faire des mathématiques, c'est aussi grâce à d'anciens professeurs. Je veux citer tout particulièrement Mme Boulais (5ème, 4ème et 2nde), Mme Deren (3ème), M. Giovendo (1ère et Terminale), Mme Monier (Math Sup), M. Mons (Math Spé 3/2), M. Monfrini et M. Castella (Télécom SudParis).

RÉSUMÉ

Dans le domaine de l'optimisation de boites noires, l'utilisateur n'a pas d'expressions analytiques de la fonction objectif et des contraintes. De fait, il n'a pas accès aux gradients. Le gradient est une information importante en optimisation étant donné qu'il permet de fournir une direction de montée de la fonction. De plus, pour récupérer les différentes valeurs de la fonction objectif et des contraintes, des simulations informatiques ou des tests en laboratoires doivent être effectués. Ceci rajoute de nombreuses difficultés supplémentaires : temps de calculs importants pour récupérer les données, bruitage des données et certaines simulations peuvent échouer.

Pour résoudre ce genre de problèmes, des algorithmes ont été développés. Parmi eux, MADS a été proposé par Audet et Dennis en 2006. C'est un algorithme itératif de recherche directe qui évalue des points de proche en proche sur un treillis. Il offrait à la base un traitement rudimentaire des contraintes, en associant une valeur infinie à tous points non-réalisables. Il a depuis été étoffé pour offrir un traitement plus souple à des contraintes de plus en plus hétéroclites.

Cette thèse propose trois nouvelles fonctionnalités à l'algorithme MADS. Premièrement, MADS calcule des modèles des contraintes afin d'ordonner les points du plus prometteur au moins prometteur. Cependant, un traitement adéquat des contraintes binaires, qui ne retournent que deux valeurs, manque dans MADS. Pour pallier cette absence, des modèles des contraintes binaires seront proposés en utilisant des outils de régression, issus de la classification supervisée.

Deuxièmement, ces mêmes outils permettent de proposer un ordonnancement nouveau des points à évaluer quand aucune fonction substitut n'est accessible dans MADS. Les points qui ont le plus de chance d'être réalisables seront évalués en premier pour favoriser la recherche de solutions réalisables de qualité. Cette stratégie sera comparée à une méthode favorisant les points les plus éloignés des points déjà évalués et à la méthode par défaut dans ce cas dans MADS, qui favorise les points qui sont le plus dans la direction du dernier succès par rapport au centre du treillis.

Enfin, il peut être noté que la mise à l'échelle des contraintes choisie par l'utilisateur au

moment de définir le problème a un impact sur le fonctionnement de MADS. MADS propose un traitement de mise à l'échelle des variables en entrée de la boite noire, mais rien pour les contraintes en sortie. Cette thèse propose une façon de les mettre à l'échelle, de sorte qu'elles prennent des valeurs de même ordre de grandeur. Cela permettra qu'elles aient globalement la même importance.

ABSTRACT

In the field of blackbox optimization, the user does not have access to the analytical expressions of the objective function and of the constraints. Thus, there is no access to the gradient. But the gradient is an important piece of information since it gives an increasing direction of the function. Moreover, in order to obtain those values, computer simulations or experiments in laboratory have to be done. This adds further difficulties: heavy computational times to get the data, noisy data and some simulations may fail.

To solve this kind of problems, algorithms have been developed. Among them, MADS has been proposed by Audet and Dennis in 2006. It is a direct search iterative algorithm that evaluates points on a mesh. At first, it offered a basic management of the constraints by associating an infinite value to all infeasible elements. Since then, more flexible ways have been proposed to handle various types of constraints. There are for example models for most of the constraints in order to sort points from the most to the least promising. However, in MADS, there is no specific management of binary constraints, which can return only two different values. Thus, models of binary constraints will be offered using tools of regression from supervised classification.

Those tools also offer new ordering methods to sort the points that need to be evaluated when no models are available in MADS. The points which are the most likely to be feasible will be evaluated first in order to look most likely for feasible solutions. This strategy will be compared to one evaluating first the elements the furthest from the ones already evaluated and to the default, in that situation, in MADS which sorts the points according to the direction of last success.

Finally, it should be pointed out that the scaling of the constraints provided by the user chosen while defining the problem has an impact on MADS's behaviour. MADS deals with the scaling of the input variables of the blackbox, but nothing is done for the constraints in the output. This thesis offers to handle the scaling of the output so that they take values of about the same range so that they have more or less the same influence.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1 INTRODUCTION

En industrie, certains problèmes d'optimisation sont suffisamment complexes pour ne pas posséder d'expressions analytiques de la fonction objectif et des contraintes. De fait, il est impossible de prévoir a priori les valeurs que vont retourner la fonction objectif et les contraintes sur un jeu de paramètres. Les domaines de définition des fonctions mises en jeu dans ce problème ne sont même pas forcément connus. L'absence de cette information peut-être vue comme une contrainte inconnue même par l'utilisateur. Cette contrainte est parfois décrite comme une «contrainte cachée». On pourrait voir cela même comme un problème qui n'est pas correctement défini par l'utilisateur. Dans ce genre de problème, pour récupérer ces valeurs, il faut réaliser une expérience, qu'elle soit informatique ou en laboratoire. Ceci est très différent d'autres domaines en recherche opérationnelle où l'accessibilité des fonctions n'est pas un problème. Pourtant, il faut mettre les efforts nécessaires pour trouver la valeur optimale de ce problème, ou du moins une valeur qui satisfasse l'utilisateur.

L'une des spécificités de ce genre de problème est l'absence d'accès à la dérivée ou au gradient. Or le gradient donne une information extrêmement utile dans la recherche d'un optimum dans un problème d'optimisation. Que ce soit par la condition d'optimalité de premier ordre, ou les conditions KKT dans le cas avec des contraintes. Le gradient fournit aussi une information précieuse sur des directions de montée et de descente et donc des directions susceptibles d'améliorer la meilleure valeur connue de la fonction objectif. Une branche de l'optimisation mathématique consiste en la création d'algorithmes n'utilisant à aucun instant le gradient, même si ce dernier peut exister. Il s'agit de «l'optimisation sans dérivées». Et dans le cadre de l'étude ici, il n'y a pas d'expressions des différentes fonctions, alors on parle «d'optimisation de boites noires» [1]. Pour résoudre ce types de problèmes d'optimisation, il existe des algorithmes itératifs qui se basent sur un maillage, qui sont appelés «algorithmes de recherche directe» pour résoudre des problèmes de boites noires. MADS («Mesh Adaptative Direct Search» ou «Recherche Directe sur Treillis Adaptatif» en français), développé en 2006 par Audet et Denis [2], est une généralisation d'un grand nombre d'entre eux. Il a connu plusieurs améliorations depuis afin de pouvoir résoudre de plus en plus de problèmes différents. Il permet notamment de résoudre des problèmes de boites noires avec des contraintes. En 2006, le traitement des contraintes est rudimentaire, via la barrière extrême [2] qui consiste à simplement associer une valeur infinie à la fonction objectif pour tous les points non-réalisables. Les années suivantes ont amené des améliorations dans leur traitement. Il y a la barrière progressive [3], plus souple que la barrière extrême car prenant en compte les valeurs prises par les

contraintes pour les points non-réalisables, l'utilisation de modèles ou de fonctions substituts pour les contraintes [4], ou même un traitement de contraintes linéaires d'égalités [5].

À chaque itération, MADS va fournir une liste de points à évaluer. Dans le cadre de la stratégie opportuniste [6], il va les évaluer un par un jusqu'à l'obtention d'un point réalisable trouvant une meilleure valeur de la fonction objectif. Les contraintes ont notamment une importance dans l'ordonnancement des points de la stratégie opportuniste. Lorsqu'il n'y a pas de modèles pour la fonction objectif et les contraintes, alors l'ordonnancement est effectué selon la direction de dernier succès. Malgré la qualité de cette méthode, il est intéressant de la questionner en proposant deux stratégies d'ordonnancement fondées sur des idées extrêmement différentes. La première est fondée sur le fait que la solution retournée est un point réalisable. L'objectif est donc d'évaluer en priorité les points dont on pense qu'ils ont plus de chances d'être réalisables. Si cette stratégie semble être sécuritaire, la seconde tentera de prendre davantage de risques en évaluant d'abord les points qui seront les plus éloignés des points déjà évalués. Cette stratégie explorera davantage des régions inconnues.

Dans l'optique de continuer dans l'amélioration du traitement des contraintes, il faut noter qu'une mise à l'échelle des valeurs des contraintes est absente du fonctionnement de MADS. D'autant plus que ceci a été pris en compte dans MADS pour les variables passées en entrée de la boite noire [7]. Un problème de mise à l'échelle se pose par exemple avec un choix d'unités, comme le fait de décider de mettre des distances en centimètres ou en kilomètres. Idéalement, un problème de mise à l'échelle dans la description du problème ne devrait pas avoir d'incidence sur le fonctionnement de MADS. Or ceci a plusieurs impacts dans MADS. Le premier est dans la fonction de violation de contraintes dans la barrière progressive. Sans mise à l'échelle, un point qui aurait été dominé pour une question d'échelle le resterait pendant tout le déroulement de l'algorithme. Il ne pourrait donc jamais être le centre d'une recherche locale. L'objectif est donc d'associer des poids aux contraintes de sorte qu'elles soient mises à l'échelle les unes par rapport aux autres. Par ailleurs, il y a des modèles qui peuvent être calculés sur les contraintes pour une stratégie d'ordonnancement dans la stratégie opportuniste. Or ces modèles peuvent perdre en fiabilité en raison d'une mise à l'échelle erronée dans la description du problème. L'étude de la mise à l'échelle dans les modèles de la fonction de violation de contraintes sera aussi étudiée dans ce manuscrit.

Même si cela n'est plus systématique, certaines contraintes sont toujours traitées par la barrière extrême. C'est le cas par exemple des contraintes binaires, c'est-à-dire des contraintes

ne retournant que 0 ou 1, indiquant si la contrainte est satisfaite ou non. En effet, le modèle quadratique [8] proposé par MADS ne peut pas convenir à ce genre de contraintes étant donné que, sauf dans de rares exceptions, les coefficients feront en sorte que le modèle quadratique prendra un ensemble non-borné de valeurs. On souhaite avoir davantage une estimation dans $[0; 1]$. Ainsi, il faut suggérer d'autres types de modèles. La classification supervisée est une branche des mathématiques qui a pour objectif de prédire la classe qu'il faut associer à un élément que l'on a pas encore évalué, sachant que les classes des éléments d'une base de données sont connues. Dans le cadre d'une contrainte binaire, les classes peuvent être les valeurs que peuvent prendre la contrainte binaire, à savoir 0 et 1. Plus encore que de prévoir la classe, on souhaite utiliser une valeur dans l'intervalle $[0; 1]$ pour pouvoir apporter davantage de finesse. Pour faire cela, il existe des méthodes de régression issues de la classification supervisée. Ainsi, ces méthodes semblent adaptées pour créer un modèle d'une contrainte binaire. Ce modèle sera par la suite utilisé dans MADS pour faciliter la stratégie d'ordonnancement quand des modèles des autres contraintes et de la fonction objectif sont également disponibles. De plus, il existe des contraintes, dites non-relaxables, que l'on ne peut jamais se permettre de violer sous peine que les autres valeurs au point évalué n'aient aucune fiabilité [9]. Elles sont, de fait, traitées par la barrière extrême. Des contraintes inconnues par l'utilisateur, appelées contraintes cachées, subissent un traitement relativement similaire, dans le sens que tout les points ne vérifiant pas les contraintes cachées sont de fait ignorés. On trouve des contraintes cachées dans certains problèmes industriels [10]. Les contraintes cachées sont souvent traitées en donnant à la fonction objectif la valeur NaN ou $+\infty$ [11]. Un traitement des contraintes non-relaxables et cachées sera proposé en les regroupant pour en faire une contrainte binaire et profiter du traitement des contraintes binaires qui sera fait.

Ce document va d'abord faire une revue de littérature de l'optimisation de boite noires, en se focalisant surtout sur les algorithmes de recherche directe, et plus spécifiquement l'algorithme MADS. Ensuite, il y aura une revue de littérature de la classification supervisée. Les chapitres suivant présenteront les travaux réalisés sur les différents projets. Un premier chapitre sur les deux nouvelles stratégies d'ordonnancement dans des problèmes sans modèles (sans accès à des fonctions substituts). Un deuxième chapitre sur la mise à l'échelle des contraintes. Un troisième chapitre se focalisant sur un traitement des contraintes binaires, non-relaxables et cachées. Ce document prendra fin avec, en conclusion, une synthèse des travaux, une limite de ces travaux, ainsi que des suggestions d'améliorations sur ces travaux et de possibles projets qui pourraient en découler.

CHAPITRE 2 REVUE DE LITTÉRATURE : OPTIMISATION DE BOITES NOIRES ET MADS

Il existe une branche de l'optimisation mathématique qui consiste à se passer du gradient, notamment quand celui-ci n'est pas accessible, que l'on appelle «optimisation sans dérivées», qui est pourtant une information extrêmement pertinente en optimisation. L'ouvrage [1] insiste d'ailleurs plusieurs fois sur l'importance de l'utiliser quand elle est accessible et fiable. Pour la suite, on définit un problème d'optimisation afin d'introduire des notations :

Soient $n, m \in \mathbb{N}$, $l, u \in \mathbb{R}^n$. Soit $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ et pour tout $\forall i \in \llbracket 1, m \rrbracket$, $c_i : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$. Considérons un problème d'optimisation de boite noire :

$$\begin{cases} \min\limits_{x \ \mathbb{R}^n} f(x) \\ \text{sous contrainte } c_i(x) \leq 0, \ \forall i \in \llbracket 1, m \rrbracket \\ \qquad\qquad\qquad l \leq x \leq u. \end{cases}$$

Remarque : Il s'agit d'un problème d'optimisation avec contrainte(s) si $m \geq 1$. Par extension, on considérera que le problème ne contient aucune contrainte si $m = 0$. Ainsi, m représente le nombre de contraintes pour ce problème.

On notera $\Omega = \{x \in \mathbb{R}^n : \forall i \in \llbracket 1, m \rrbracket, \ c_i(x) \leq 0\}$ le domaine réalisable.

L'optimisation de boite noire est une sous-branche de l'optimisation sans dérivées. Outre l'absence de dérivées, une boite noire se caractérise par une absence totale d'information, notamment concernant l'expression analytique de la fonction objectif et des possibles contraintes, et évidemment le fait de n'avoir aucune information utilisable du gradient.

S'il existe une information partielle sur le problème, alors le terme de «boite grise» est parfois utilisé. Cette information peut par exemple être de nature à donner des précisions sur la monotonie de la fonction objectif et des contraintes [12, 13]. Cependant, ce cadre ne sera pas étudié ici.

En pratique, il y a deux grandes familles de problèmes qui répondent à cette problématique de boites noires : des simulations informatiques et des tests en laboratoire. Les deux cas sont assez différents. Pour certains tests en laboratoire, il n'y a pas de modèle assez précis, et cela nécessite donc une expérience industrielle réelle pour récupérer les valeurs de la fonction objectif et des contraintes. Pour les simulations informatiques, il s'agit de programmes informatiques, souvent lourds en temps de calcul, et dont il est difficile de deviner la sortie. Ce n'est pas toujours une boite noire à proprement parler, car dans certains cas, le code est accessible pour la lecture, mais on peut la considérer comme telle en décidant de ne pas lire le code. Cependant, ces deux cas différents ont des points communs concernant notamment les difficultés qu'ils engendrent.

Outre le fait de ne pas avoir les expressions analytiques des fonctions mises en jeu par le problème d'optimisation associé, le temps de calcul pour récupérer les valeurs de ces fonctions en un point donné est parfois long. Cela peut s'évaluer en secondes, en minutes, voire en jours. Les valeurs sont récupérées sans savoir comment elles ont été calculées, tel que schématisé par la figure 2.1.

Figure 2.1 Fonctionnement d'une boite noire.

Ces longs temps de calcul mènent à davantage de difficultés. La première est que le temps de calcul important oblige à limiter le nombre d'évaluations. Il est souvent de l'ordre de grandeur de quelques milliers (des budgets de 1500 évaluations sont octroyés dans les résultats numériques de [14] et [8] sur des problèmes avec des boites noires). Par ailleurs, beaucoup d'outils de comparaison d'algorithmes pour les problèmes de boites noires n'utilisent pas le temps (typiquement en secondes) comme pour des problèmes de théorie des graphes [15] ou des problèmes de clustering [16], mais le nombre d'évaluations. En effet, en pratique, dans les problèmes d'optimisation de boites noires, le temps de traitement des informations fournies

par la boite noire est négligeable par rapport à l'évaluation de celle-ci.

Pour résoudre ce genre de problèmes, plusieurs algorithmes se passant de dérivées ont été développés. Parmi eux, il y a notamment l'algorithme de Nelder-Mead se basant sur un simplexe suivant certaines règles d'évolutions [17]. Il a été initialement conçu pour des problèmes sans contraintes. Cet algorithme fait ses comparaisons principalement par rapport à un autre algorithme sans dérivées créé par Powell en 1964 [18]. Une généralisation de l'algorithme de Nelder-Mead a été proposée dans un contexte d'optimisation sans dérivées en présence de contraintes d'inégalités [14]. À noter que dès 1965, Fletcher s'est intéressé à regrouper des méthodes basées sur l'optimisation sans dérivées [19]. Des algorithmes génétiques ont aussi été utilisés [20, 21]. Bien que le gradient ne soit pas une information accessible, il existe des algorithmes qui proposent une approximation du gradient pour l'exploiter comme la «descente basée sur le modèle» («Model-Based Descent» en anglais) [1].

Il y a différentes applications des problèmes d'optimisation sans dérivées et d'optimisation de boites noires. Il y a, parmi d'autres, des applications en énergie [22–26], en chimie [27–30], en aéronautique [31–33], dans le domaine médical [34–36], en électronique [37], dans un contexte d'eaux contaminés [38], en positionnement de détecteurs [39] ou encore en paramétrisation [40, 41].

Pour la suite, des algorithmes de «recherche directe» vont être étudiés car l'algorithme MADS, qui est celui qui nous intéressera le plus, est la généralisation de plusieurs algorithmes de cette famille. À l'inverse de certains des algorithmes présentés ci-dessus (comme l'algorithme de Nelder-Mead [17] ou les algorithmes génétiques), ces algorithmes ont une analyse de convergence et ainsi proposent des solutions avec des propriétés sous certaines conditions.

2.1 Recherche par coordonnées

L'algorithme de recherche par coordonnées (CS) est l'un des premiers algorithmes utilisés pour résoudre les problèmes sans dérivées et sans contraintes. Il est pour la première fois évoqué dans [42]. Le terme de «compass search» (recherche par boussole) est parfois utilisé [43]. Un résumé court consisterait à dire que, comme son nom l'indique, il s'agit d'une méthode itérative où chaque point évalué a été obtenu à partir du point précédent en ne modifiant qu'une seule de ses coordonnées. Il est l'ancêtre d'un bon nombre d'autres algorithmes d'op-

timisation sans dérivées comme GPS («General Pattern Search» ou «Recherche par motif généralisé» en français, voir la section 2.2) ou MADS («Mesh Adaptative Direct Search» ou «Recherche directe à maillage adaptatif» en français, voir la section 2.3).

Soient $n \in \mathbb{N}$, $f : \mathbb{R}^n \mapsto \mathbb{R}$ et considérons le problème d'optimisation sans contraintes

$$\min_{x \ \mathbb{R}^n} f(x).$$

Voici une description de cet algorithme. On notera $\mathcal{B} = (e_i)_{i \ \llbracket 1;n \rrbracket}$ la base canonique de $\mathbb{R}^n$. On se fixe un point de départ $x^0 \in \mathbb{R}^n$. On dispose d'un ensemble de sonde, ou «treillis», autour de x^0, d'un certain pas noté $\delta^0 > 0$, $P_0 = \{x^0 \pm \delta^0 e_i, i \in \llbracket 1; n \rrbracket\}$. On évalue les éléments de P_0. Si l'on a un voisin $x^1 \in P_0$ qui améliore la valeur de f, alors on crée un treillis de même pas autour de x^1 et on recommence (figure 2.2) avec le même pas $\delta^1 = \delta^0$. Sinon, on reste autour de x^0 mais on diminue la taille du pas (figure 2.3) en suivant la règle $\delta^1 = \tau\delta^0$ avec $\tau \in \mathbb{Q} \cap]0; 1[$. L'algorithme se termine à une itération $k \in \mathbb{N}$ quand le treillis a un pas δ^k inférieur à un certain seuil $\varepsilon \in \mathbb{R}_+$.

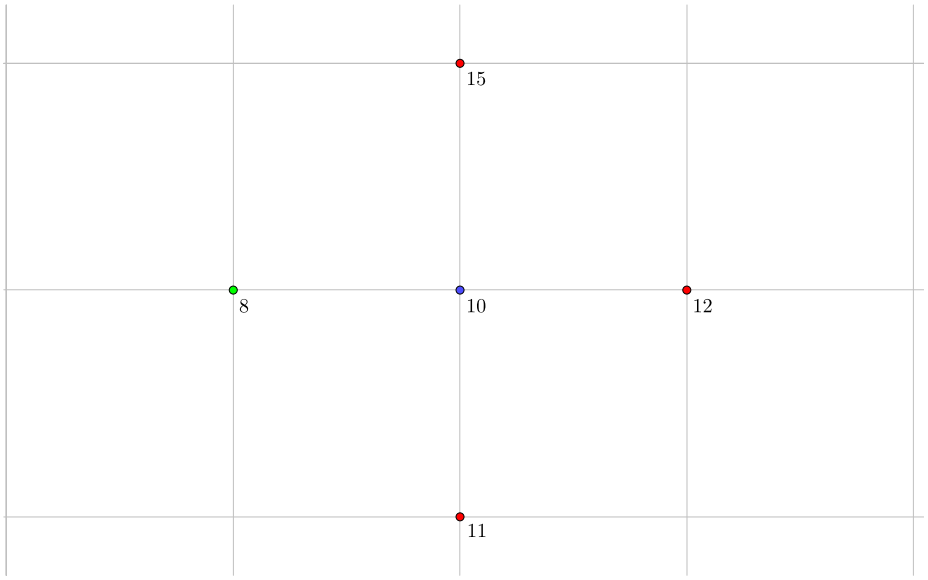


Figure 2.2 Cas où l'on a une amélioration.

L'algorithme 1 décrit le fonctionnement général de la recherche par coordonnées.

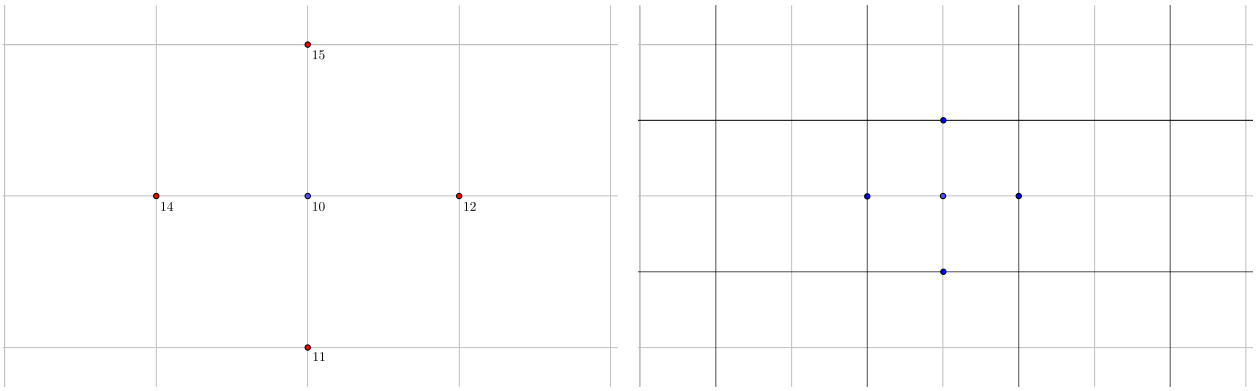


Figure 2.3 En cas d'échec, diminution du pas du maillage.

Algorithme 1 : Recherche par coordonnées

Paramètres : $x^0 \in \mathbb{R}^n$,$\Delta^0 \in \mathbb{R}_+$, $\varepsilon \in \mathbb{R}_+$, $k = 0$, $\tau \in \mathbb{Q} \cap]0;1[$
while $\delta^k \geq \varepsilon$ **do**
  if $\min\{f(x) | x = x^k \pm \delta^k e_i, i \in \mathtt{J1;}n\mathtt{K}\} < f(x_k)\}$ **then**
    $x^{k+1} \in \arg\min\{f(x), x = x^k \pm \delta^k e_i, i \in \mathtt{J1;}n\mathtt{K}\}$
    $\delta^{k+1} = \delta^k$
  else
    $x^{k+1} = x^k$
    $\delta^{k+1} = \tau\delta^k$
  end if
  $k = k + 1$
end while

Lorsque l'on diminue le pas du treillis, c'est que le voisinage de x^k n'améliore pas la fonction objectif. On dit alors que x^k est un «optimum local du treillis».

Analyse de convergence (tirée de [1]) Il faut étudier les caractéristiques de cet algorithme, notamment ses propriétés si on laisse l'algorithme faire une infinité d'itérations, lorsque $\varepsilon = 0$. Cette hypothèse est faite dans toute l'analyse de convergence. Pour cela, il faut rappeler d'abord la notion d'«ensemble de niveau».

Définition 2.1. *Soit $r \in \mathbb{R}$. On appelle ensemble de niveau r l'ensemble des éléments de $\mathbb{R}^n$ qui prennent une valeur inférieure ou égale à r par f : $\{x \in \mathbb{R}^n : f(x) \leq r\}$.*

À partir de cette définition, il existe un théorème concernant la convergence du pas du treillis dans la recherche par coordonnées.

Théorème 2.1. *Si f est une fonction dont les ensembles de niveaux sont bornés alors la suite des pas du treillis $\{\delta^k\}_{k \in \mathbb{N}}$ converge vers 0 : $\lim_{k \to +\infty} \delta^k = 0$.*

Il en découle du théorème 2.1 que si les hypothèses de ce théorème sont vérifiées et que le critère d'arrêt $\delta^k \geq \varepsilon$ est tel que si ε est strictement positif, alors l'algorithme se termine en temps fini.

La démonstration proposée dans [1] montre d'ailleurs que si les ensembles de niveaux de f sont bornés, alors la suite d'itérés $\{x^k\}_{k \in \mathbb{N}}$ contient une infinité de minimums locaux sur le treillis. Notons $K \subset \mathbb{N}$ les itérations ayant fourni un minimum local sur le treillis, et donc provoqué une diminution du pas du treillis. De plus, cette démonstration montre que $\{x^k\}_{k \in \mathbb{N}}$ est inclus dans l'ensemble de niveau $f(x^0)$ qui est borné, d'où de même pour $\{x^k\}_{k \in K}$. Il en résulte que $\{x^k\}_{k \in K}$ est inclus dans l'adhérence de l'ensemble de niveau $f(x^0)$ qui est donc un fermé borné d'un espace vectoriel de dimension finie, donc est un compact. $\{x^k\}_{k \in K}$ étant une suite incluse dans un compact, il existe $K' \subset K$ tel que $\{x^k\}_{k \in K'}$ converge. Comme $K' \subset K$, alors $\{x^k\}_{k \in K'}$ est une suite de minimums locaux du treillis convergente. Une telle suite existe donc. Cette suite permet d'introduire le prochain théorème.

Théorème 2.2. *Si f est une fonction continue dont les ensembles de niveaux sont bornés et soit $\hat{x}$ la limite d'une sous-suite convergente de minimums locaux du treillis produite par la recherche par coordonnées, alors pour tout $d \in \{\pm e_i : i \in \mathbb{N}\}$, alors, soit $f'(\hat{x}; d) \leq 0$, soit $f'(\hat{x}; d)$ n'existe pas. Si, en outre, f est C^1, alors $\nabla f(\hat{x}) = 0$.*

Cependant, malgré cette analyse de convergence, la recherche par coordonnées n'est pas sans faille. Ceci va être mis en évidence dans le prochain paragraphe.

Limite de la recherche par coordonnées Certains exemples mettent à mal l'algorithme de recherche par coordonnées. On peut trouver des exemples où la recherche par coordonnées telle qu'elle a été présentée ne permet pas de converger vers un optimum local. Un exemple est la minimisation de la norme infinie [1].

$$\min_{x \ \mathbb{R}^2} f(x), \text{ avec } f(x) = ||x||_\infty .$$

Prenons comme point de départ le point $x^0 = (1;1)$ et un maillage de taille $\Delta^0 = 1$ et $\tau = \frac{1}{2}$. Les voisins sont donc $(1;0)$, $(0;1)$, $(2;1)$, $(1;2)$. On a $f(x^0) = 1$. Or $f(0;1) = 1$, $f(1;0) = 1$, $f(2;1) = 2$, $f(1;2) = 2$. Aucun des voisins de x^0 n'améliore la valeur de f. On diminue donc le pas du treillis par 2. Les nouveaux voisins de x^0 dans ce nouveau maillage sont donc $(0.5;1)$, $(1;0.5)$, $(1.5;1)$, $(1;1.5)$.

x	$f(x)$
$(0.5;1)$	1
$(1;0.5)$	1
$(1.5;1)$	1.5
$(1;1.5)$	1.5

On observe que l'on n'améliore pas le meilleur itéré courant x_0, donc le pas du treillis se réduit et le centre du treillis reste x_0.

En procédant ainsi, on pourrait montrer par récurrence qu'après $q \in \mathbb{N}$ itérations, l'itéré x^q serait toujours en $x^q = x^0$, que le pas du maillage serait de $\frac{1}{2^q}$, que les voisins de x^0 et leurs valeurs par f dans ce maillage seraient donc :

x	$f(x)$
$(1 ; 1 - \frac{1}{2^q})$	1
$(1 - \frac{1}{2^q} ; 1)$	1
$(1 ; 1 + \frac{1}{2^q})$	$1 + \frac{1}{2^q}$
$(1 + \frac{1}{2^q} ; 1)$	$1 + \frac{1}{2^q}$

On observe encore qu'aucun point n'améliore la valeur de f, donc le pas du treillis se réduit et le centre du treillis reste x_0.

Ainsi, à chaque itération, aucun des voisins ne fournit une amélioration : le pas est donc diminué et on reste au même point x^0. L'algorithme converge donc vers le point x^0. Or ce point n'est pas un minimum local pour ce problème puisque le seul minimum local est le point $(0;0)$, qui est même le minimum global. L'algorithme ne converge donc pas nécessairement vers un minimum local, même si la fonction objectif est convexe.

Cette faiblesse découle du fait que les directions sont fortement restreintes. Ce sont toujours les mêmes directions avec différentes normes. Si il y avait eu la possibilité d'aller dans la direction $-e_1e_2$, il y aurait pu y avoir une amélioration de x^0.

2.2 Recherche par motifs généralisée

L'algorithme de recherche par motifs généralisée (General Pattern Search ou GPS dans la littérature) est une généralisation de l'algorithme de recherche par coordonnées. Il a été décrit par Torczon [44]. Il a pour principal objectif d'atténuer le problème de sous-représentation des directions.
À chaque itération $k \in \mathbb{N}$, on va se donner un ensemble de directions D_k parmi un ensemble de directions unitaires D déterminé au début. On souhaite que D forme une base positive.

Définition 2.2. $\{v_1, \dots, v_l\} \subset \mathbb{R}^n$ *est une base positive si* $\left\{\sum_{i=1}^{l} \lambda_i v_i : \lambda_i \geq 0\right\} = \mathbb{R}^n$ *et si* $\sum_{i=1}^{l} \lambda_i v_i = 0$, $\lambda_i \geq 0$ *implique pour tout* $i \in \mathtt{J}1, l\mathtt{K}$ *que* $\lambda_i = 0$.

L'article [45] prouve qu'une base positive contient entre $n+1$ et $2n$ éléments. Ainsi, D contient entre $n+1$ et $2n$ directions unitaires.

L'algorithme permet également une phase d'exploration (nommée en anglais «search»). Il s'agit d'une phase optionnelle, qui permet à l'utilisateur d'évaluer une liste de points selon sa propre stratégie. Cette phase a plusieurs avantages. La recherche locale se contentait d'explorer une seule région dans le but de trouver un minimum local. Avec une phase d'exploration, on peut explorer d'autres régions dans l'espoir de trouver un meilleur minimum local, voire un minimum global. Pour cela, l'utilisateur peut fournir une liste finie de points qui sont

sur le treillis. On peut citer par exemple la possibilité d'utiliser un hypercube latin (dont on trouve une description dans l'article [46]), une recherche à voisinage variable («Variable Neighborhood Search» en anglais ou «VNS», décrit par Mladenović et Hansen [47]) ou une recherche de type Nelder-Mead [14], ou une fonction substitut. Le nouvel algorithme devient alors l'algorithme 2.

Algorithme 2 : Recherche par motifs

Paramètres : $x^0 \in \mathbb{R}^n$,$\delta^0 \in \mathbb{R}_+$, $\varepsilon \in \mathbb{R}_+$, $k = 0$, $\tau \in \mathbb{Q} \cap]0;1[$, $w^+ \in \mathbb{N}$, $w^- \in \mathbb{Z}_-$, $D \subset \mathbb{R}^n$.
while $\Delta^k \geq \varepsilon$ **do**

1 : Exploration (optionnelle)

Évaluer une liste finie S_k de points sur le treillis.

2 : Sonde locale

Générer les directions $D_k \subseteq D$
Évaluer f sur l'ensemble $P_k = \{x^k \pm \delta^k d : d \in D_k\}$

3 : Mise à jour

if On trouve, dans la phase 1 ou 2, $x \in \mathbb{R}^n$ tel que $f(x) < f(x^k)$ **then**
  $\delta^{k+1} = \tau^{w_k}\delta_k$ avec $w_k \in \llbracket w^-, 0 \rrbracket$.
  $x^{k+1} \in \arg\min\{f(x) : x = x^k \pm \delta^k d, d \in D_k\}$
else
  $x^{k+1} = x^k$
  $\delta^{k+1} = \tau^{w_k}\delta^k$ avec $w_k \in \llbracket 1, w^+ \rrbracket$.
end if
$k = k + 1$
end while
Retourner en phase 1.

On remarque qu'il suffit de choisir pour tout $k \in \mathbb{N}$, $D_k = \{\pm e_i, i \in \llbracket 1, n \rrbracket\}$ et $S_k = \emptyset$ afin d'obtenir l'algorithme de recherche par coordonnées.

Gradient généralisé et dérivée de Clarke En optimisation de boite noire, la différentiabilité d'une fonction ne peut pas être vérifiée. C'est souvent une hypothèse trop forte par rapport à la situation réelle. On cherche donc une généralisation de la notion de différentiabilité qui prolongerait cette notion. Clarke, dans son ouvrage [48], regroupe les définitions et les principaux résultats du gradient généralisé et de la dérivée de Clarke.

Définition 2.3. *On appelle gradient généralisé d'une fonction* $f : \mathbb{R}^n \mapsto \mathbb{R}$ *au point* $x \in \mathbb{R}^n$ *dans la direction* $d \in \mathbb{R}^n$ *la limite, si elle existe :*

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

Ce genre d'outils permet également de caractériser un minimum local.

Théorème 2.3. *$\hat{x} \in \mathbb{R}^n$ est un minimum local d'une fonction f localement lipschitzienne autour du point $\hat{x}$ ssi pour tout $d \in \mathbb{R}^n$, $f^\circ(\hat{x};d) \geq 0$.*

Une autre façon de caractériser $f^\circ(x;d) \geq 0$ est

Propriété 2.1. *Soit $x \in \mathbb{R}^n$ et f une fonction localement lipschitzienne autour de x. Alors, $f^\circ(x;d) \geq 0 \Leftrightarrow 0 \in \partial f = \{s \in \mathbb{R}^n | \forall v \in \mathbb{R}^n, f^\circ(x;d) \geq v^T s\}$*

Ainsi, on a pu abaisser les conditions d'optimalité de différentiable à localement lipschitzienne. La dérivée de Clarke sera utilisée dans l'analyse de convergence de l'algorithme de recherche par motif.

Analyse de convergence L'analyse de convergence de GPS provient de [44] et [49] et est faite avec les hypothèses suivantes : $\varepsilon = 0$, pour tout $k \in \mathbb{N}$, S_k est fini et inclus dans le treillis et la suite des itérés $\{x^k\}_{k \in \mathbb{N}}$ est incluse dans un ensemble compact. On supposera dans le reste de la section que la suite des itérés produits par GPS est incluse dans un ensemble compact. Cette dernière hypothèse est garantie si les ensembles de niveaux de f sont compacts.

L'analyse de convergence commence d'abord par un théorème concernant le pas du treillis $\{\delta^k\}_{k \in \mathbb{N}}$.

Théorème 2.4. *Le pas du treillis vérifie la relation* $\liminf_{k \to +\infty} \delta^k = 0$.

Il est intéressant de noter qu'à l'inverse de la recherche par coordonnées, il s'agit d'une limite inférieure au lieu d'une simple limite. Ceci est dû au fait que GPS permet l'augmentation du pas du treillis δ^k. L'analyse se poursuit avec l'étude des itérés $\{x^k\}_{k \in \mathbb{N}}$.

Définition 2.4. *Soit $K \subseteq \mathbb{N}$. On dit que la sous-suite d'itérés de minimums locaux du treillis $\{x^k\}_{k \in K}$ est une sous-suite raffinante si elle converge et si $\{\delta^k\}_{k \in K}$ converge vers 0.*

Le prochain résultat garantit l'existence d'une telle sous-suite.

Théorème 2.5. *Pour toute instance de GPS, et sous les hypothèses décrites ci-dessus, il existe une sous-suite raffinante convergente.*

Démonstration : Soit K l'ensemble des indices des itérés qui ont été des minimum locaux du treillis. Alors, par le théorème 2.4, il existe $K \subset K$ tel que $\lim_{k \in K} \delta^k = 0$. De plus, la sous-suite $\{x^k\}_{k \in K}$ est incluse dans un compact. Il existe donc $K \subset K$ tel que la sous-suite $\{x^k\}_{k \in K}$ est convergente. De plus, comme $K \subset K$ alors $\lim_{k \in K} \delta^k = 0$.

On voit donc que les sous-suites raffinante sont construites à partir d'itérations qui ont provoqué des diminutions du treillis. On s'intéresse maintenant à la qualité de la solution obtenue après une infinité d'itérations.

Théorème 2.6. *Soient $\{x^k\}_{k \in K}$ une sous-suite raffinante qui converge vers $\hat{x}$ et $d \in D$ tel que pour toute itération $k \in K$ de la sonde, f ait été évalué dans la direction d et si f est localement lipschitzienne autour de $\hat{x}$, alors $f^\circ(\hat{x}; d) \geq 0$. De plus, comme f est localement lipschitzienne en $\hat{x}$ alors elle est continue en $\hat{x}$ donc $\lim_{k \in K} f(x^k) = f(\hat{x})$. Si de plus f est strictement différentiable, alors $\nabla f(\hat{x}) = 0$.*

Ce dernier résultat apporte donc également une garantie sur la valeur de f obtenue. Cependant, le fait que le nombre de directions D soit fini limite le théorème 2.6. On souhaite donc avoir un algorithme n'ayant pas une telle restriction au niveau des directions.

GSS Avant d'évoquer MADS, il est possible de parler de l'algorithme de «recherche par ensemble générateur» («Generating Set Search» en anglais, ou GSS) [50]. La différence majeure concerne le choix du centre de la recherche locale. Il change si on a trouvé à l'itération courante $k \in \mathbb{N}$ un point $x \in \mathbb{R}^n$ réalisable qui bat notre centre courant $x_k \in \mathbb{R}^n$ en obtenant une meilleure valeur par f d'une marge $\rho(\delta_k)$. Ainsi, on pose $x_{k+1} = x$ si et seulement si $f(x) < f(x_k) - \rho(\delta_k)$.

2.3 Descriptif de MADS

L'algorithme MADS est une généralisation de l'algorithme de recherche par motifs généralisée proposée en 2006 [2]. Il y a donc des similitudes dans l'analyse de convergence de MADS avec celle de la recherche par motifs généralisée. Il a connu plusieurs développements informatiques, notamment sur Matlab et en Python, mais surtout en C++ [51].

Soient $n, m \in \mathbb{N}$, $X \subset \mathbb{R}^n$, $l, u \in \mathbb{R}^n$. Soit $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ et pour tout $i \in \llbracket 1, m \rrbracket$, $c_i : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$. Considérons un problème d'optimisation de boite noire :

$$\begin{cases} \min\limits_{x \ X} f(x) \\ \text{sous contrainte } c_i(x) \leq 0, \ \forall i \in \llbracket 1, m \rrbracket \\ \qquad\qquad\qquad l \leq x \leq u. \end{cases}$$

$\Omega = \{x \in X : \forall i \in \llbracket 1, m \rrbracket, \ c_i(x) \leq 0\}$ est le domaine réalisable.

X contient l'ensemble des contraintes non-relaxables, qui sont des contraintes que l'on ne peut pas violer car sinon les autres valeurs n'ont aucune légitimité. Les contraintes $c_i(x) \leq 0$, pour tout $i \in \llbracket 1, m \rrbracket$ sont dites relaxables ; elles peuvent ne pas être satisfaites tout en conservant la légitimité des autres valeurs de la boite noire [9]. l et u définissent les bornes des variables.

Comme pour GPS, il y a une phase d'exploration relativement libre pour l'utilisateur et une phase de sonde qui garantit la convergence. MADS a pour objectif de proposer des directions encore plus diversifiées que la recherche par motifs dans sa phase de sonde. Pour cela, on ne disposera pas d'un seul pas de maillage mais de deux : δ^k, qui est le pas du maillage, et Δ^k, le pas de la sonde. La recherche par motifs généralisée correspondrait au cas où $\delta^k = \Delta^k$. L'intérêt d'avoir deux pas est de proposer, lorsque l'on affine le maillage, plus de directions possibles.

À l'itération $k \in \mathbb{N}$, la cache, c'est-à dire la liste des points évalués depuis le début de la résolution par MADS au début de cette itération, est notée $\mathcal{C}^k$. Le treillis est décrit dans [2] à l'aide de $D \in \mathbb{R}^{n \times n_D}$ de sorte que D soit un ensemble générateur positif et tel qu'il existe $G \in \mathbb{R}^{n \times n}$ inversible et $Z \in \mathbb{R}^{n \times n_D}$ avec $D = GZ$. Le treillis est défini alors par

$$M_k = \{x + \delta_k Dz : z \in \mathbb{N}^{n_D}, x \in \mathcal{C}_k\}.$$

Il s'agit en fait d'une union de treillis, de sorte que chaque élément de la cache soit sur le treillis. La recherche locale s'effectue cependant autour d'un seul point. L'ensemble de sonde est décrit dans [2] par $P_k = \{x^k + \delta^k d : d \in D_k\}$ avec $D_k \in \mathbb{R}^{n \times n_D}$ est un ensemble générateur positif tel que pour élément d de D_k :

— Pour $d \neq 0$ il existe $u \in \mathbb{N}^{n_D}$, dépendant de l'itération k, tel que $d = Du$.

— $\Delta^k ||d|| \leq \delta^k \max\{||d|| : d \in D\}$.

— La limite, au sens de «limite de bases positives ordonnées» définie dans [52] des éléments normalisés de D_k est un ensemble générateur positif.

Comme pour GPS, en cas d'échec à cette itération, on diminue la taille des deux pas, et en cas d'une amélioration, on change de centre de recherche locale et on augmente les deux pas. La figure 2.4 montre une configuration de réduction de maillage après un échec. L'algorithme 3 décrit l'algorithme MADS.

Figure 2.4 Diminution du pas de maillage en cas d'échec.

2.3.1 Stratégie opportuniste et fonctions substituts

Sans la stratégie opportuniste, à chaque itération $k \in \mathbb{N}$, MADS fait évaluer tous les points d'une liste de points $\mathcal{L}_k$. Cette liste $\mathcal{L}_k$ correspond à $\mathcal{S}_k$ lors d'une phase d'exploration et correspond à $\mathcal{P}_k$ lors d'une phase de sonde. Dans l'optique de préserver le budget d'évaluations et d'en éviter certaines inutiles, on souhaite profiter des points qui ont fourni un succès, c'est-à-dire des points réalisables qui ont obtenu une meilleure valeur par f. Le centre de la recherche locale change pour aller vers une région prometteuse. Une façon de procéder est d'utiliser la stratégie opportuniste. La stratégie opportuniste consiste à arrêter l'évaluation des points de $\mathcal{L}_k$ dès l'obtention du premier succès [2] et ainsi de mettre fin à l'itération de la phase d'exploration ou de la phase de sonde locale dans 3.

Définition 2.5. *À l'itération $k \in \mathbb{N}$ dont le meilleur itéré au début de l'itération est $x_k \in \Omega \subset \mathbb{R}^n$, on dit que le point $x \in \mathcal{L}_k$ est un succès si x est un point réalisable avec une*

Algorithme 3 : MADS

Paramètres : $x^0 \in \mathbb{R}^n$,$\Delta^0 = \delta^0 = 1 \in \mathbb{R}_+$, $\varepsilon \in \mathbb{R}_+$, $k = 0$, $\tau \in \mathbb{Q} \cap]0;1[$, $w^+ \in \mathbb{N}$, $w^- \in \mathbb{Z}_-$, D tel que décrit auparavant.
while $\Delta^k \geq \varepsilon$ **do**

1 : Exploration (optionelle)

Évaluer une liste finie S_k de points sur le treillis.

2 : Sonde locale

Générer les directions D_k afin de constituer un ensemble générateur positif comme décrit auparavant.
Évaluer f sur l'ensemble $P_k = \{x^k + \delta^k d : d \in D_k\}$ de façon opportuniste.

3 : Mise à jour

if On trouve, dans la phase 1 ou 2, $x \in \mathbb{R}^n$ tel que $f(x) < f(x^k)$ **then**
$x^{k+1} = x$.
$\Delta^{k+1} = \tau^{w_k} \Delta^k$ avec $w_k \in \llbracket w^-, 0 \rrbracket$.
$x^{k+1} \in \arg\min\{f(x) : x = x^k \pm \Delta^k d, d \in D_k\}$.
else
$x^{k+1} = x^k$.
$\Delta^{k+1} = \tau^{w_k} \Delta^k$ avec $w_k \in \llbracket 1, w^+ \rrbracket$.
end if
Choisir δ^{k+1} tel que $\delta^{k+1} < \Delta^{k+1}$.
$k = k + 1$.
end while

meilleure valeur de f*, c'est-à-dire* $x \in \Omega$ *et* $f(x) < f(x^k)$.

De plus, dès que l'on obtient un succès, le point issu de ce succès devient le nouveau centre de notre recherche locale.

Afin de profiter au maximum de la stratégie opportuniste, on souhaite trouver, s'il existe, le plus rapidement, un point de $\mathcal{L}_k$ fournissant un succès. Plus un succès est trouvé rapidement parmi $\mathcal{L}_k$, plus on passe rapidement à l'itération suivante et donc plus le budget d'évaluation est préservé. Dès lors, l'ordre dans lequel les points $\mathcal{L}_k$ sont évalués a de l'importance [6]. Selon l'ordre dans lequel les points de $\mathcal{L}_k$ sont présentés à la boite noire, la performance de la stratégie opportuniste peut être très affectée. On souhaite faire un ordonnancement qui augmente les chances de trouver un succès rapidement. L'algorithmique de la stratégie opportuniste est décrite dans l'algorithme 4.

Algorithme 4 : Évaluations avec stratégie opportuniste

Entrées : $k \in \mathbb{N}$, $\mathcal{L}_k \subset \mathbb{R}^n$, x^k le meilleur itéré courant.

for $x \in \mathcal{L}_k$ **do**
  if $x \in \Omega$ et $f(x) < f(x^k)$ **then**
    $x^{k+1} = x$
    STOP
  end if
end for

Pour déterminer l'ordre dans lequel les points sont évalués, on souhaite faire un pré-traitement des éléments de $\mathcal{L}_k$ avant de les évaluer avec la boite noire. La façon de choisir un ordre pour $\mathcal{L}_k$ s'appelle «une stratégie d'ordonnancement». Le pré-traitement peut se faire via l'estimation de f et des contraintes définissant Ω pour chaque élément de $\mathcal{L}_k$ et d'un ordonnancement en tenant compte de ces estimations.

L'une des stratégies d'ordonnancement est de passer par des fonctions substituts, notamment de f et des contraintes. Les fonctions substituts peuvent être des problèmes simplifiés du problème de base, fournis par un expert du problème. Ces substituts prennent moins de temps à calculer, afin de fournir un ou plusieurs points intéressants à évaluer. Les fonctions substituts peuvent aussi être définies à l'aide de modèles. En 1999, plusieurs auteurs ont proposé un travail autour des fonctions substituts, avec entre autres des exemples de l'industrie [53]. Les

applications sont nombreuses, comme par exemple dans [54]. Déjà évoquées précédemment dans 2.2, des modèles quadratiques peuvent être utilisés [8], ou encore des bibliothèques de logiciels comme «R» avec sa bibliothèque «dynaTree» [55]. Pour quantifier la performance de fonctions substituts par rapport à d'autres, des outils ont été développés afin de les comparer [56].

Pour que l'ordonnancement soit intéressant, il faut que les temps de calculs des fonctions substituts soient négligeables par rapport à l'évaluation de la boite noire. Ceci est souvent le cas car les temps de calculs sont en général longs pour les «vraies» boites noires, tels que celles issues d'expériences industrielles ou de certains codes informatiques. Une itération de MADS avec la stratégie opportuniste est décrite par l'algorithme 5.

Algorithme 5 : Itération avec la stratégie opportuniste

Entrées : $k \in \mathbb{N}$, x^k le meilleur itéré courant.

MADS génère une liste de points $\mathcal{L}_k$ à évaluer.
Ordonner les éléments de $\mathcal{L}_k$.
Evaluer les éléments de $\mathcal{L}_k$ en suivant la stratégie opportuniste
STOP

Ceci pouvant être résumé par la figure 2.5

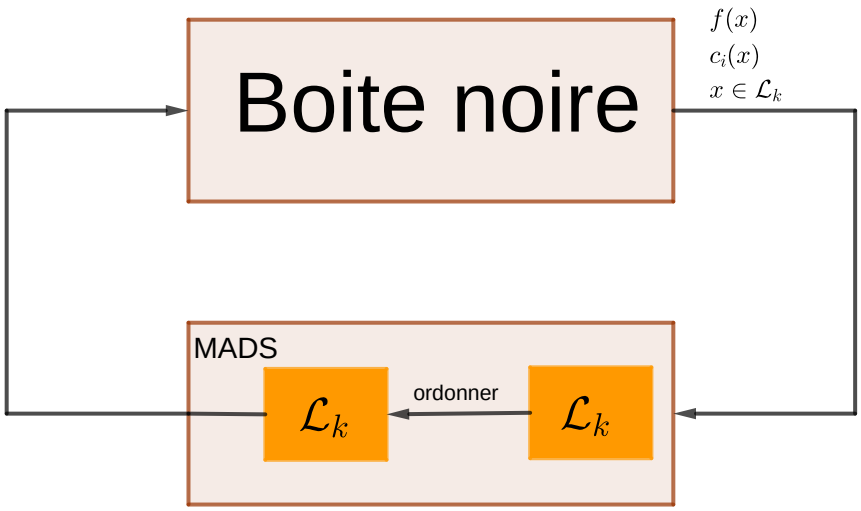


Figure 2.5 Stratégie d'ordonnancement dans MADS.

2.3.2 Traitement des contraintes

En optimisation de boites noires, Le Digabel et Wild ont proposé dans [9] de classifier les différents types de contraintes dans la littérature, de donner une description à chacune et de leur donner un nom. Dans les chapitres 5 et 6, de nouvelles façons de traiter les contraintes vont être exposées concernant la mise à l'échelle des contraintes et le traitement des contraintes binaires, non-relaxables et cachées.

Barrière extrême La barrière extrême [2] est une méthode de résolution qui consiste à résoudre le problème d'optimisation $\min_{x \ \mathbb{R}^n} f_\Omega(x)$

où pour tout $x \in \mathbb{R}^n$, $f_\Omega(x) = \begin{cases} f(x) \text{ si } x \in \Omega \\ +\infty \text{ si } x \notin \Omega \end{cases}$.

L'avantage de cette transformation est de reformuler un problème d'optimisation avec contraintes en un problème d'optimisation sans contraintes mais il requiert un point de départ $x_0 \in \Omega$ réalisable. Il peut être souligné que les contraintes cachées (celles qui sont inconnues de l'utilisateur) peuvent être traitées par cette approche [11].

Barrière progressive La barrière progressive a ensuite été proposée [3]. L'idée est de ne pas rejeter définitivement les points non-réalisables et d'utiliser les informations qu'ils

apportent. Pour quantifier la violation des contraintes, une fonction a été définie. Un élément $x \in \mathbb{R}^n$ peut ne pas vérifier une contrainte non-relaxable de X. Dans ce cas, la fonction de violation des contraintes assignera à l'infini pour $x \notin X$.

Définition 2.6. *On appelle h la fonction de violation des contraintes comme suit :*

$$h(x) = \begin{cases} \sum_{i=1}^{m} \max(0, c_i(x))^2 & \text{si } x \in X \\ +\infty & \text{sinon.} \end{cases}$$

L'utilisation de l'exposant 2 dans le terme $\max(0, c_i(x))^2$ a été choisie pour préserver la régularité des fonctions c_i. Il découle de façon immédiate de la définition 2.6 la propriété suivante :

Propriété 2.2. *Soit $x \in \mathbb{R}^n$ $h(x) = 0$ si, et seulement si, x est réalisable par rapport à $\Omega = \{x \in X : \forall i \in \llbracket 1, m \rrbracket, c_i(x) \leq 0\}$.*

On notera $V_k = \mathcal{C}_k \bigcap X$, l'ensemble des points déjà évalués au début de l'itération k qui vérifient toutes les contraintes non-relaxables.

On représente chaque point $x \in V_k$ de la cache dans un plan avec en abscisse la valeur $h(x)$ et en ordonnée la valeur $f(x)$. Tous les points x de la cache se retrouvant sur l'axe des ordonnées dans ce plan vérifient $h(x) = 0$ et sont donc réalisables.

On peut considérer notre problème d'optimisation comme un problème bi-objectif où nos deux fonctions sont f et h. Soit $F_k = \underset{x \ \mathcal{C}_k}{\arg\min}\{f(x) : h(x) = 0\}$ l'ensemble des meilleurs points réalisables. Dans la barrière progressive, les seuls points réalisables à être conservés sont ceux avec la plus faible valeur de f. On récupère la meilleure valeur de f parmi les points réalisables avec

$$f_k^F = \begin{cases} +\infty & \text{si } F_k = \emptyset \\ f(x) & \text{pour n'importe quel } x \in F_k \text{ sinon.} \end{cases}$$

Parmi les points non-réalisables, il faut créer une relation d'ordre partiel et conserver les points non-dominés au regard de cette relation d'ordre.

Définition 2.7. *Soient $x, y \in V_k$, alors on notera $x \prec y$ si $h(x) < h(y)$ et $f(x) \leq f(y)$ ou si $h(x) \leq h(y)$ et $f(x) < f(y)$. On dira dans ce cas que x domine y.*

Parmi tous les points de V_k non-réalisables, on ne conservera que ceux qui ne sont pas dominés. Pour cela, soit $U_k = \{x \in V_k - \Omega : \nexists y \in V_k - \Omega, y \prec x\}$.

On fournit en plus un paramètre positif h^k_{max} à chaque itération k. Plutôt que de rejeter tous les points non-réalisables, on va rejeter tous les points de U_k qui violent les contraintes de façon trop importante, à savoir quand la violation dépasse le paramètre h^k_{max}. On va rejeter tous les points de la cache x tels que $h(x) > h^k_{max}$. La barrière progressive conserve tous les points non-réalisables de l'ensemble $I_k = \underset{x \ U_k}{\arg\min}\{f(x) : 0 < h(x) < h^k_{max}\}$.

À partir de I_k, on définit le couple (h^I_k, f^I_k) qui permettra la mise à jour, notamment, de h^k_{max} :

$$(h^I_k, f^I_k) = \begin{cases} (+\infty, +\infty) & \text{si } I_k = \emptyset \\ (h(x), f(x)) & \text{pour n'importe quel } x \in I_k \text{ sinon.} \end{cases}$$

On va s'intéresser à la construction de $(h^k_{max})_{k \ \mathbb{N}}$. Il y a trois types d'itérations : dominantes, améliorantes et échouantes.

— Une itération $k \in \mathbb{N}$ est dite dominante s'il existe $y \in V_{k+1}$ tel que :
$h(y) = 0$ et $f(y) < f^F_k$, ou $h(y) > 0$ et $y \prec x$ pour tout $x \in I_k$.

— Une itération $k \in \mathbb{N}$ est dite améliorante s'il existe $y \in V_{k+1}$ non-réalisable tel que :
$0 < h(y) < h^I_k$ et $f(y) > f^I_k$.

— Une itération $k \in \mathbb{N}$ est dite échouante si pour tout $y \in V_{k+1}$:
$h(y) = 0$ et $f(y) \geq f^F_k$, ou $h(y) > 0$ et $f(y) \geq f^I_k$, ou $h(y) > h^I_k$.

On met à jour par

$$h^{k+1}_{max} = \begin{cases} \max\limits_{y \ V_{k+1}} \{h(y) : h(y) < h^I_k\} & \text{si l'itération } k \text{ est améliorante} \\ h^I_k & \text{sinon.} \end{cases}$$

Les règles de mise à jour de h^I_k n'ont pas été évoquées, mais elles permettent, cumulées avec

celles de h^k_{max}, de garantir que la suite $\{h^k_{max}\}_{k \in \mathbb{N}}$ est décroissante [3]. De plus, on sait que cette suite est minorée par 0, donc la suite converge.

Il est à noter que la barrière progressive a également été déployée dans un contexte de régions de confiance [57].

Autres Entre les deux, la «barrière progressive à extrême» (*progressive-to-extreme barrier* dans sa présentation dans [58]) consiste à traiter une contrainte relaxable tant qu'elle n'est pas réalisable avec la barrière progressive. Une fois que l'on a un point qui vérifie la contrainte, elle est traitée avec la barrière extrême. Un développement a aussi été effectué pour des contraintes linéaires d'égalité explicites [5].

2.3.3 Convergence de MADS

Une analyse de convergence de MADS apparait dès [2]. Comme pour GPS, l'analyse de convergence s'effectue en étudiant l'algorithme 3 avec $\varepsilon = 0$ et pour tout $k \in \mathbb{N}$, S_k est fini et inclus sur le treillis. Une même hypothèse que pour GPS est faite pour l'analyse de convergence de MADS. On supposera dans cette section que la suite des itérés produits par MADS est incluse dans un ensemble compact. Une fois de plus, cette hypothèse est garantie si les ensembles de niveaux de f sont compacts. La convergence de l'algorithme passe d'abord par la convergence des différents pas.

Propriété 2.3. *Pour toute instance de MADS,* $\liminf_{k \to +\infty} \Delta^k = 0$ *et* $\liminf_{k \to +\infty} \delta^k = 0$.

La notion de sous-suite raffinante vue pour la recherche par motifs généralisée est donc similaire. Pour GPS, il fallait que δ^k converge vers 0. Pour MADS, on voudrait que les deux pas convergent vers 0 pour pouvoir utiliser cette notion, ce qui est le cas grâce à la proposition précédente. L'existence d'une telle sous-suite lorsque les itérés sont compris dans un ensemble borné est assurée pour MADS, comme elle l'était pour GPS.

Cette notion de sous-suite raffinante permet également de définir les directions raffinante.

Définition 2.8. *Soit* $K \subset \mathbb{N}$ *et* $\{x^k\}_{k \in K}$ *une sous-suite raffinante convergeant vers* $\hat{x}$. d *est appelé «direction raffinante de* $\hat{x}$*» si et seulement si il existe* $L \subset K$ *dénombrable et des directions* $d^k \in D_k$ *tels que* $x^k + \delta^k d^k \in \Omega$ *et* $\lim_{k \in L} \frac{d^k}{\|d^k\|} = \frac{d}{\|d\|}$.

Parmi ces directions, celles qui sont dans les cônes hypertangents de Ω sont celles qui seront étudiées pour la dérivée directionelle généralisée en $\hat{x}$. La définition est issue de [2], mais une définition équivalente a été présentée d'abord dans [59].

Définition 2.9. *Soient $x, v \in \mathbb{R}^n$ et $S \subseteq \mathbb{R}^n$. On dit que v est un vecteur tangent de Clarke à S au point x si, et seulement si, pour toute suite $\{y^k\}_{k \in \mathbb{N}} \in S^{\mathbb{N}}$ qui converge vers x et pour toute suite $\{t_k\}_{k \in \mathbb{N}} \in (R_+)^{\mathbb{N}}$ qui converge vers 0, il existe une suite, $\{w_k\}_{k \in \mathbb{N}} \in (\mathbb{R}^n)^{\mathbb{N}}$ convergeant vers v telle que, pour tout $k \in \mathbb{N}$, $y^k + t_k w_k \in S$.*
L'ensemble des vecteurs tangents de Clarke à S au point x forme un ensemble appelé cône tangent de Clarke et est noté $T_S^{Cl}(x)$.

Il en découle le résultat suivant :

Théorème 2.7. *Soient $\hat{x}$ la limite d'une sous-suite raffinante et $d \in T_\Omega^H(\hat{x})$ une direction raffinante de $\hat{x}$. Supposons que f est lipschitzienne en $\hat{x}$. Alors $f^\circ(\hat{x}; d) \geq 0$.*

Il reste à savoir quelles vont être les directions raffinante qui vont se trouver dans T_Ω^H. Un ensemble de directions obtenu à l'aide de la transformation de Householder a été utilisé dans MADS d'abord dans l'article [60]. À chaque itération, la matrice de Householder $H^k = I - 2v^k(v^k)^T$ est créée à partir d'un vecteur unitaire v^k (I est la matrice identité et v^k est un vecteur colonne). L'ensemble des colonnes de H^k, noté $\mathcal{H}^k$ est utilisé pour générer les directions à l'itération k.

Théorème 2.8. *Soient $\{v^k\}_{k \in \mathbb{N}}$, pour tout $k \in \mathbb{N}$, H^k la matrice de Householder construite à partir de v^k et $\mathcal{H}^k$ l'ensemble des colonnes de H^k. Alors, si $\{v^k, k \in \mathbb{N}\}$ est dense dans la sphère unité, alors $\bigcup_{k \in \mathbb{N}} H^k$ est asymptotiquement dense dans la sphère unité.*

Ce théorèeme est issu du livre [1]. Il précise qu'en construisant ses directions de la sorte, MADS construit un ensemble de directions asymptotiquement dense.

2.3.4 Mise à l'échelle dynamique

Dans la version de MADS précédemment décrite, le pas du maillage est le même selon toutes les directions. Or, les variables peuvent être de nature très différentes et donc à des ordres de grandeurs très différents. Dans [7], une nouvelle version de MADS est proposée avec des pas de maillage qui varient selon les différentes variables.

Soit $\in \mathbb{N}$, le numéro de l'itération courante de MADS. Pour tout $j \in [1, n]$, soient $\delta_j > 0$ le pas du maillage de la variable j à l'itération et $\Delta_j > 0$ le pas de sonde de la variable j à

l'itération . Le fonctionnement de MADS est le même hormis que le pas du maillage Δ est remplacé par Δ_j, pour tout $j \in \mathtt{J1}, n\mathtt{K}$ et le pas de la sonde δ est remplacé par δ_j, pour tout $j \in \mathtt{J1}, n\mathtt{K}$.

L'analyse de convergence est globalement la même que celle décrite par [2].

NOMAD Il existe plusieurs développements informatiques de MADS pour pouvoir résoudre des problèmes d'optimisation de boites noires faits sous le nom de «NOMAD». Il existe notamment un développement en C++ [61] dont le fonctionnement est décrit dans l'article [62]. Un guide pour les utilisateurs de cette version de NOMAD a été créé pour rendre son utilisation plus simple et comprendre les différents paramètres [51]. Par ailleurs, il existe des interfaces à divers langages, comme MATLAB, PYTHON et R.

2.4 Profils de données et de performances

Pour comparer différentes méthodes, notées $\mathcal{M}$, une série de graphiques a été proposée, appelés «profils». Ils sont basés sur la capacité des différents algorithmes à résoudre une série de problèmes. On se donne un ensemble de problèmes de minimisation $\mathcal{P}$. Pour un problème dont la fonction objectif est f, le test de convergence dépend de x_0 (le point de départ réalisable) et de f_L, la meilleure valeur trouvée par tous les algorithmes sur ce problème. On dit qu'un algorithme résout le problème à une précision $\tau > 0$ si l'algorithme a généré un point x tel que

$$f(x_0) - f(x) \leq (1 - \tau)(f(x_0) - f_L).$$

Ce test de convergence étant défini, il est possible de définir les profils de performance et de données. Deux types de profils vont être présentés.

Profil de performance En 2002, Dolan et Moré proposent les «profils de performance» [63]. La mesure de la performance est le nombre d'unités de mesure nécessaire pour vérifier le critère de convergence avec une précision $\tau > 0$. Elle peut s'effectuer selon le choix de l'utilisateur. Cette mesure peut être le temps utilisé par le processeur (en secondes par exemple) ou le nombre d'évaluations. Pour un problème $p \in \mathcal{P}$ et une méthode $M \in \mathcal{M}$, cette mesure sera notée $t_{p,M}$. Si la méthode M ne parvient pas à atteindre le critère de convergence avec

une précision $\tau > 0$ pour le problème p, alors la convention est que $t_{p,M} = +\infty$.

Le ratio de performance est défini par

$$r_{p,M} = \frac{t_{p,M}}{\min\{t_{p,M} : M \in \mathcal{M}\}} \in [1; +\infty[.$$

La meilleure méthode pour le problème p aura un ratio de performance valant 1. Les méthodes qui ne satisferont pas le critère de performance auront un ratio infini.

Les profils de performance établissent le ratio de problèmes pouvant être résolus (c'est-à-dire vérifiant le critère de convergence) à la précision τ parmi ceux de $\mathcal{P}$ avec un ratio de performance inférieur à $\alpha \geq 1$. Ce ratio s'écrit, pour $m \in \mathcal{M}$,

$$\rho_M(\alpha) = \frac{1}{Card(P)} Card(\{p \in \mathcal{P} : r_{p,M} \leq \alpha\}).$$

Le profil de performance consiste à faire une représentation graphique, pour tout $m \in \mathcal{M}$ de la fonction ρ_M. Il faut noter que le critère de convergence dépend de la précision τ. Il y a donc un profil de performance par précision.

Profil de données En 2009, Moré et Wild proposent les «profils de données» [64]. Pour les profils de données, $t_{p,M}$ doit représenter le nombre d'évaluations. Les profils de données évaluent, pour une méthode $M \in \mathcal{M}$, le ratio de problèmes résolus à la précision τ en moins de α évaluations. On définit

$$d_M(\alpha) = \frac{1}{Card(\mathcal{P})} Card(\{p \in \mathcal{P} : t_{p,M} \leq \alpha\}).$$

Le profil de données consiste à faire une représentation graphique, pour tout $M \in \mathcal{M}$, de la fonction d_M. Cependant, le profil de données ne dépend pas de la dimension du problème. Or il est logique que si la dimension d'un problème est plus grande, il faudra un nombre d'évaluations plus important pour résoudre ce problème.

Il existe une autre version du profil de données prenant en compte la dimension $n_p \in \mathbb{N}$ du problème p. Il s'agit de voir quel ratio de problèmes est résolu à la précision τ en moins de $\alpha(n_p + 1)$ évaluations. Le facteur $n_p + 1$ est lié au fait qu'il faille $n_p + 1$ points pour faire une estimation du gradient. Le profil de données peut donc consister également à représenter,

pour tout $M \in \mathcal{M}$, la fonction d_M mais avec

$$d_M(\alpha) = \frac{1}{Card(\mathcal{P})} Card(\{p \in \mathcal{P} : t_{p,M} \leq \alpha(n_p + 1)\}).$$

Une synthèse de ces profils, ainsi qu'une description d'autres outils pour comparer des algorithmes d'optimisation, est effectuée dans [65].

CHAPITRE 3 CLASSIFICATION SUPERVISÉE

La classification supervisée est une branche des mathématiques qui a pour objectif de classer des éléments dans des classes connues à l'avance. À titre d'exemples de classification supervisée, on pourra citer la reconnaissance des chiffres manuscrits [66] (les classes sont dès lors les chiffres de 0 à 9) ou la prédiction de résultats sportifs [67] (victoire de l'équipe qui joue à domicile, match nul ou victoire de l'équipe qui joue à l'extérieur).

Dans la classification supervisée, on dispose d'un ensemble d'entraînement, où chacun des éléments est associé à une classe. La prédiction de la classe d'un élément qui n'est pas présent dans l'ensemble d'entraînement est forcément l'une des classes d'un des éléments de cet ensemble. On ne prédit pas de nouvelle classe.

La prédiction peut varier selon la méthode de classification supervisée choisie. Certaines méthodes reposent sur des modèles, linéaire par exemple pour la régression logistique [68] ou l'analyse discriminante linéaire (bien que cette dernière puisse être généralisée à l'aide de noyaux [69,70]). D'autres, comme la méthode des k-plus-proches-voisins (k-nn), ne requièrent aucune connaissance a priori sur les données [71].

Ce chapitre propose d'étudier quelques outils utilisés en classification supervisée. Il ne s'agit pas d'une revue de littérature exhaustive, mais le but est de décrire les outils de classification adéquats pour les chapitres 4 et 6. Dans le cadre de ce travail portant sur l'optimisation de boites noires, tous les problèmes étudiés possèdent deux classes : d'un côté les éléments réalisables (classe 0) et de l'autre les éléments non-réalisables (classe 1). L'objectif sera de prédire si un point, ne figurant pas dans la cache, a plus de chances d'être réalisable ou pas. Ceci permettra de pouvoir ordonner plusieurs points selon ce critère, du plus prometteur au moins prometteur.

On appellera «candidat» tout élément dont on voudra prédire la classe. On notera $\mathcal{C}$ l'ensemble des points de la base de données (pour coïncider avec la cache de la boite noire) et $N \in \mathbb{N}$ sa cardinalité. Les éléments de $\mathcal{C}$ seront notés $\{x^1, \dots, x^N\} \subseteq \mathbb{R}^n$. Pour tout $i \in \llbracket 1; N \rrbracket$, la classe associée à l'élément x^i est notée $b(x^i) \in \{0; 1\}$ (en référence notamment aux notations du chapitre 6 avec les contraintes binaires). On notera, pour le candidat $x \in \mathbb{R}^n$, $\tilde{b}(x) \in [0; 1]$, la probabilité d'appartenir à la classe 1 faite par une méthodes (comme

l'une parmi celles décrites dans ce chapitre), et $\tilde{cl}(x) \in \{0; 1\}$ l'estimation de la classe associée.

Dans toutes méthodes de classification supervisée, il faut déterminer des paramètres afin de pouvoir prédire les classes d'éléments non-évalués. Cette phase s'appelle «phase d'apprentissage» ou «phase d'entrainement». Une mesure de l a qualité d'une méthode de classification supervisée, via les paramètres mis en jeu, dépend de sa capacité à prédire la classe d'éléments qui n'ont pas encore été évalués.

Surapprentissage L'introduction du livre «Pattern Classification» de Duda, Hart et Stork [72] montre un exemple avec la prédiction sur le fait qu'un poisson soit un saumon ou un bar. Elle montre un modèle qui sépare parfaitement les saumons et les bars lors de la phase d'apprentissage. Mais pour cela, le modèle faisait tout pour que des éléments qui semblaient être des anomalies, comme des bars qui ne ressemblaient pas aux autres bars mais qui ressemblaient plutôt à des saumons, soient classés correctement. Mais ce modèle sera pourtant un modèle de mauvaise qualité car il ne donnera pas de bonnes prédictions sur des poissons qui n'ont pas servi à la phase d'apprentissage. Cette introduction montre ensuite un modèle plus simple, qui contient des éléments utilisés en phase d'entrainement qui sont mal classés. Pourtant il semble mieux correspondre à la réalité. Le fait d'avoir un apprentissage des données qui classe très bien les données utilisées en phase d'apprentissage mais mal les éléments non évalués encore est appelé «surapprentissage».

Il est donc important de tester les méthodes à l'aide d'éléments qui n'ont pas été utilisés en phase d'apprentissage. La validation croisée est un outil de sélection de méthodes qui a notamment pour but d'éviter le surapprentissage.

3.1 Validation croisée

La validation croisée est un outil qui permet la sélection d'une méthode ou d'un algorithme [73] dans une certaine liste, qui sera notée $\mathcal{M}$. Elle repose sur le principe d'un ensemble d'entrainement et d'un ensemble d'évaluation. L'une des versions les plus connues est la « -fold-cross-validation» [1].
Cela consiste à diviser $\mathcal{C}$ (l'ensemble des points évalués jusqu'alors) en sous-ensembles de

1. ℓ sera utilisé afin de ne pas confondre avec le k de k-nn plus tard. Dans la littérature, on trouve souvent k-fold-cross-validation.

tailles sensiblement égales ($\left\lfloor \frac{|\mathcal{C}|}{l} \right\rfloor$ ou ($\left\lfloor \frac{|\mathcal{C}|}{l} \right\rfloor + 1$). Parmi les différentes versions, celle qui sera choisie sera la «leave-one-out-cross-validation» (validation croisée sauf un), qui sera notée LOOCV qui est le cas particulier de la « -fold-cross-validation» en prenant $= |\mathcal{C}|$. En prenant la base de données, on retire un élément $x \in \mathbb{R}^n$. On effectue ensuite l'estimation $\tilde{b}_M(x)$ que la méthode M aurait faite de l'élément retiré si on n'avait pas évalué x auparavant. On compare ensuite avec la vraie valeur $b(x)$. L'algorithme 6 présente ce cas là.

Algorithme 6 : «Leave-One-Out-Cross-Validation» (LOOCV)

---

1 : Initialisation

Pour chaque méthode $M \in \mathcal{M}$, initialiser $Err(M) = 0$.

Entrées : La cache $\mathcal{C}$, une liste de méthodes $\mathcal{M}$.

1 : Tester les algorithmes

Pour chaque $x \in \mathcal{C}$ faire l'estimation $\tilde{b}_M(x)$ à l'aide de la méthode M en utilisant $\mathcal{C} - \{x\}$ comme ensemble d'entrainement.

Poser $Err(M) \leftarrow Err(M) + (b(x) - \tilde{b}_M(x))^2$.

2 : Sélection

Choisir $M \in \arg\min_M Err(M)$.

3.2 Présentation de quelques méthodes de classification supervisée

3.2.1 k-plus-proches-voisins (k-nn)

La méthode des k-plus-proches-voisins, souvent utilisée sous le nom k-nn, est une méthode de classification supervisée se basant principalement sur la distance du candidat aux autres éléments de la base de données. Sa simplicité lui a permis d'avoir de nombreuses variantes, dont certaines sont décrites dans [74]. Le but de k-nn est de trouver les éléments de $x^1, \ldots, x^N \in \mathcal{C}$ qui sont les plus proches de $x^0 \in \mathbb{R}^n$ (point dont on souhaite estimer la classe), en terme de distance. Si ces éléments sont les plus proches en terme de distance, on peut penser qu'ils devraient potentiellement avoir des ressemblances sur d'autres aspects, notamment sur leurs classes. C'est une méthode qui est peu sensible aux données bruitées [74].

Le choix d'une classe s'effectue d'abord par le calcul de la probabilité d'appartenir à la classe 1. Plus la valeur de cette régression sera proche de 0, plus il y a de raisons de penser que la classe associée a de chances d'être 0. À l'inverse, plus cette valeur sera proche de 1, plus il y a de raisons de penser que la classe associée a de chances d'être 1. La classe d'un élément $x^i, i \in \llbracket 1; N \rrbracket$ sera notée $b(x^i)$. Pour faire cette estimation, on effectue une moyenne pondérée des classes des k plus proches voisins de x^0 [75].

$$\tilde{b}(x^0) = \frac{1}{k} \sum_{i=1}^{k} \frac{w_i b(x^i)}{\sum_{j=1}^{k} w_i}.$$

Cette pondération peut d'effectuer par un noyau positif. Un noyau est une fonction $K : \mathbb{R} \mapsto \mathbb{R}$ telle que $\int_{\mathbb{R}} K(u)\,\mathrm{d}u = 1$. Un noyau positif K est un noyau qui vérifie en plus, pour tout $u \in \mathbb{R}$, $K(u) \geq 0$. Des pondérations avec des poids qui ne sont pas issus d'un noyau est aussi possible [76, 77].

Une fois le noyau choisi, la pondération s'effectue par

$$\tilde{b}(x^0) = \frac{1}{k} \sum_{i=1}^{k} \frac{K\left(\frac{||x^0 - x^i||}{h}\right) b(x^i)}{\sum_{j=1}^{k} K\left(\frac{||x^0 - x^j||}{h}\right)}.$$

avec $h > 0$, qui est un réel appelé «bande passante».

Le noyau fait intervenir une norme. Celle qui sera choisi dans le reste du document est la norme euclidienne. Elle possède l'avantage d'être invariante par rotation.

Choix du noyau Plusieurs choix de noyaux sont possibles. Parmi les plus courants, on trouve :

— $K(u) = \frac{1}{2a} \mathbb{1}_{[-a,a]}(u)$, $a \in \mathbb{R}_+$. On rappelle que si A est un sous-ensemble de $\mathbb{R}$, la fonction $\mathbb{1}_A$ est définie par, pour tout $u \in \mathbb{R}$,

$$\mathbb{1}_A(u) = \begin{cases} 1 & \text{si } u \in A \\ 0 & \text{si } u \notin A. \end{cases}$$

C'est ce que l'on appelle le noyau uniforme. Son impact sur la pondération est de

donner la même importance à chacun des voisins. On effectue donc un vote parmi les voisins en regardant la proportion d'éléments de classes 0 et 1. La classification peut donc s'effectuer en prenant la classe la plus représentée parmi les voisins à l'aide du noyau uniforme. Cependant, ce noyau ne tient pas compte des distances. Le voisin le plus proche a autant d'importance que le voisin le plus éloigné.

— $K(u) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{u^2}{2\sigma^2})$, $u \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$ est le noyau gaussien. Il prend en considération les distances. De plus, son support est $\mathbb{R}$, donc si on prend $k = N$ voisins, alors tous les éléments de l'ensemble d'entraînement ont un impact sur la classification.

— $K(u) = \begin{cases} \frac{3}{4}(1-u^2) & \text{si } u \in [-1,1] \\ 0 & \text{sinon} \end{cases}$ est le noyau d'Epanechnikov [78].

Ce dernier noyau minimise l'Erreur Quadratique Intégrale Moyenne Asymptotique ($AMISE$) [79] et donc profite d'une optimalité selon ce critère. $AMISE$ est un développement asymptotique de $MISE$ [80, 81]

$$MISE = \int_{-\infty}^{+\infty} E((b(x) - \tilde{b}(x))^2)\, \mathrm{d}x.$$

L'expression de $AMISE$, comme montrée dans [80] utilise la dérivée seconde de la fonction b, ce qui n'est pas réellement possible dans notre cadre, étant donné qu'elle est non-dérivable dès qu'elle ne prend pas qu'une seule valeur. La théorie de $MISE$ et $AMISE$ est surtout réalisée dans un contexte d'estimation d'une densité de fonction de probabilité.

Les noyaux sont souvent choisis décroissants sur $\mathbb{R}_+$ (si $0 < a \leq b$, alors $K(a) \geq K(b)$), pour que les éléments les plus éloignés aient le moins d'impact sur la classification.

3.2.2 Analyse discriminante linéaire

L'analyse discriminante linéaire [72, 82] est une méthode de projection des données selon un axe. Une fois que l'on a déterminer cette projection, on va effectuer une classification à l'aide de cette même projection. Il est utile de préciser que, si un élément est défini comme étant dans $\mathbb{R}^n$, alors il est défini comme un vecteur colonne.

Soient $\mathcal{C}_0$ l'ensemble des éléments de classe 0 et $\mathcal{C}_1$ l'ensemble des éléments de classe 1. On va noter $m_0 \in \mathbb{R}^n$ (resp. $m_1 \in \mathbb{R}^n$) le centre de gravité des éléments de $\mathcal{C}_0$ (resp. de $\mathcal{C}_1$) et $m \in \mathbb{R}^n$ le centre de gravité des éléments de $\mathcal{C}$.

On va noter, pour $i \in \{0;1\}$, $S_i = \sum_{x \ C_i} (x - m_i)(x - m_i)^T$. La matrice $S_W = S_0 + S_1$ est la matrice de dispersion intra-classe. On appelle $S_B = (m_0 - m_1)(m_0 - m_1)^T$ la matrice de dispersion inter-classe. C'est une matrice $n \times n$ de rang 1. Par construction, S_0, S_1, S_B et S_W sont toutes semi-définies positives.

On cherche $w \in \mathbb{R}^n$, un vecteur unitaire, de façon à minimiser $w^T S_W w$ et à maximiser $w^T S_B w$. Plutôt que de résoudre un problème bi-objectif, on va plutôt chercher $w \in \mathbb{R}^n$ unitaire qui maximise le critère de Fisher [82] : $\frac{w^T S_B w}{w^T S_W w}$. Sans perte de généralité, on peut choisir la norme de w de telle sorte que $w^T S_W w = 1$ au lieu de prendre w unitaire. Le problème se formule dès lors par :

$$\begin{cases} \min\limits_{w \ \mathbb{R}^n} w^T S_B w \\ \text{sous contrainte } w^T S_W w = 1 \end{cases}$$

Le lagrangien de ce problème est $L(w, \lambda) = w^T S_B w - \lambda(w^T S_W w - 1)$. On cherche les points critiques de L.

$$\begin{aligned} \tfrac{\partial L}{\partial w}(w, \lambda) = 0 &\iff S_B w - \lambda S_W w = 0 \\ &\iff S_W^{-1} S_B w = \lambda w \text{ (à condition que } S_W \text{ soit inversible)} \\ \tfrac{\partial L}{\partial \lambda}(w, \lambda) = 0 &\iff -(w^T S_W w - 1) = 0 \\ &\iff w^T S_W w = 1 \end{aligned}$$

Ceci implique la résolution de :

$$\begin{cases} S_W^{-1} S_B w = \lambda w \\ w^T S_W w = 1 \end{cases}$$

sous condition que S_W soit inversible.

Remarque : L'inversibilité de S_W peut poser problème. Il peut arriver que l'une des colonnes de S_W soit nulle. Ceci arrive notamment lorsque tous les éléments de $\mathcal{C}$ ont une composante qui est la même. Si tous les éléments ont une composante qui est la même, alors cette com-

posante n'apporte pas d'information. Elle peut donc être retirée Dans ce qui suit, on suppose que S_W est inversible et donc définie positive.

On doit donc résoudre un problème de valeur propre. En effet λ est donc une valeur propre de $S_W^{-1}S_B$ et w est un vecteur propre associé à cette valeur propre. Supposons, que l'on ait toutes ces valeurs propres, alors pour toute valeur propre λ de $S_W^{-1}S_B$, $S_W^{-1}S_B w = \lambda w$. D'où $S_B w = \lambda S_W w$ et donc $w^T S_B w = \lambda w^T S_W w$ et enfin $\frac{w^T S_B w}{w^T S_W w} = \lambda$. Or, $\frac{w^T S_B w}{w^T S_W w}$ est la quantité que l'on souhaite maximiser. La solution de notre problème initial est λ_{max}, la plus grande valeur propre de $S_W^{-1}S_B$ et w un vecteur propre associé à λ_{max}. Or, par construction, S_B est de rang 1 (si $m_0 \neq m_1$), donc $S_W^{-1}S_B$ est de rang 1. Ainsi 0 est valeur propre de $S_W^{-1}S_B$ d'ordre $(n-1)$. De plus, en posant $w = S_W^{-1}(m_0 - m_1)$, alors :

$$\begin{aligned} S_W^{-1}S_B w &= S_W^{-1}S_B S_W^{-1}(m_0 - m_1) \\ &= S_W^{-1}(m_0 - m_1)(m_0 - m_1)^T S_W^{-1}(m_0 - m_1) \text{ (par définition de } S_B). \end{aligned}$$

Or, $(m_0 - m_1)^T S_W^{-1}(m_0 - m_1) \in \mathbb{R}$, donc :

$$\begin{aligned} S_W^{-1}S_B w &= (m_0 - m_1)^T S_W^{-1}(m_0 - m_1) S_W^{-1}(m_0 - m_1) \\ &= (m_0 - m_1)^T S_W^{-1}(m_0 - m_1) w \end{aligned}$$

On remarque que $(m_0 - m_1)^T S_W^{-1}(m_0 - m_1)$ est la dernière valeur propre recherchée et $w = S_W^{-1}(m_0 - m_1)$ est le vecteur propre associé. Or, S_W est définie positive, donc la valeur propre $(m_0 - m_1)^T S_W^{-1}(m_0 - m_1)$ est strictement positive. C'est donc la solution de notre problème. Il n'est donc pas nécessaire de calculer les valeurs propres ; le vecteur selon laquelle la projection va s'effectuer est $w = S_W^{-1}(m_0 - m_1)$.

Pour effectuer la classification d'un élément $x \in \mathbb{R}^n$, m_0, m_1 et x sont projetés sur le sous-espace vectoriel engendré par le vecteur w . On dira que x est de classe 0 si son projeté est plus proche du projeté de m_0 que du projeté de m_1. La figure 3 de [82] représente cette projection dans le cas de trois classes (dont le fonctionnement est similaire). C'est donc une classification basée sur un critère purement géométrique.

Il est également possible de faire de la régression [83]. Cela requiert cependant de poser certaines hypothèses. Il faut que la distribution des éléments d'une classe donnée suive une loi normale multivariée et que les matrices de covariances soient égales.

Cas non linéaire Il est possible d'adapter l'analyse discriminante linéaire pour faire une classification non-linéaire. Cette méthode consiste, comme nous le verrons pour les SVM en section 3.2.5, à placer les éléments de la base de données dans un espace de dimension plus importante à l'aide d'une méthode par noyau [69, 84].

3.2.3 Régression logistique

On souhaite faire une régression avec une fonction qui prend ses valeurs dans $[0; 1]$. Une fonction souvent choisie est la fonction logistique qui donne par ailleurs son nom à cette méthode. La fonction logistique est, pour $x^i = [x_1^i, \ldots, x_n^i] \in \mathbb{R}^n$:

$$f(x^i) = \frac{\exp\left(\beta_0 + \sum_{k=1}^{n} \beta_k x_k^i\right)}{1+\exp\left(\beta_0 + \sum_{k=1}^{n} \beta_k x_k^i\right)}$$

où $\beta_0, \ldots, \beta_n \in \mathbb{R}$ sont $n+1$ paramètres qu'il nous faudra identifier [85]. Les β_k, $k \in \llbracket 0; n \rrbracket$ sont stockés dans une matrice $\beta = [\beta_0, \ldots, \beta_n]^T$.

Une fois que β est déterminé, la régression sera effectuée par [75] :

$$\tilde{b}(x^i) = \frac{\exp\left(\beta_0 + \sum_{k=1}^{n} \beta_k x_k^i\right)}{1+\exp\left(\beta_0 + \sum_{k=1}^{n} \beta_k x_k^i\right)}$$

ou encore :

$$\tilde{b}(x^i) = \frac{\exp\left(\tilde{x^i}\beta\right)}{1+\exp\left(\tilde{x^i}\beta\right)}$$

avec $\tilde{x^i} = [1, x_1^i, \ldots, x_n^i]$.

Il est intéressant de noter que $\ln(\frac{\tilde{b}(x^i)}{1-\tilde{b}(x^i)}) = \tilde{x^i}\beta$.

Le critère utilisé pour l'estimation de notre probabilité, et donc des coefficients de β, est le maximum de vraisemblance. Chaque variable est une variable de Bernoulli. Après calcul, on trouve que la log-vraisemblance vaut

$$l(\beta, \mathcal{C}) = \sum_{i=1}^{N} \beta \tilde{x^i} b(x^i) - \sum_{i=1}^{N} \ln\left(1 + \exp\left(\beta \tilde{x^i}\right)\right) \tag{3.1}$$

On recherche le point critique de la log-vraisemblance. Or, il n'existe pas de réponse analytique à ce problème. Il faut donc estimer β de façon numérique. Historiquement, la méthode la plus populaire est la méthode de Newton-Raphson [86].

Il faut que les données ne soient pas linéairement séparables pour avoir un bon fonctionnement de la régression logistique. Sinon, la recherche du maximum de vraisemblance pose plusieurs problèmes développés dans [87, 88].

Bien que le maximum de vraisemblance soit souvent utilisé dans ce cadre, les paramètres de β ont tendance à être surestimés. Pour répondre à ce défaut, il est possible de chercher plutôt un maximum de vraisemblance pénalisée [89].

3.2.4 Réseaux de neurones

La méthode par réseaux de neurones est une méthode de classification et de régression qui est inspirée du fonctionnement du cerveau et sa capacité à analyser ce qui l'entoure. Elle permet notamment de faire du traitement non-linéaire du problème [90]. Le cerveau dispose de neurones et il existe des connexions entre ces neurones. Ces connexions permettent de transmettre de l'information. Un neurone peut recevoir de l'information de plusieurs autres neurones et traite l'information en conséquence. Une fois traitée, cette information se transmet aux autres neurones de façon plus complexe, etc.

Le réseau est en couches, où les éléments de la couche du dessous transmettent les informations aux éléments de la couche du dessus. Nous allons présenter principalement un réseau de neurones en trois couches, celui-ci résumant assez bien le fonctionnement des réseaux de neurones. La suite reprend principalement les explications de [75] et de [72].

Réseaux de neurones à trois couches Avec un réseau de neurones à trois couches, on va avoir une couche d'entrée où l'on aura n neurones. Chaque neurone comprend chaque composante de l'entrée $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Les informations de la couche d'entrée vont propager l'information vers une couche dite «cachée», qui va elle même propager l'information vers la couche de sortie. Ces deux dernières propagations vont s'effectuer à l'aide de paramètres

décidés par une phase d'apprentissage que nous étudierons plus tard. C'est la couche de sortie qui va nous aider à prendre la décision. Par exemple, avec la régression, on aurait un seul neurone dans la couche de sortie et la valeur contenue dans ce neurone contient la valeur souhaitée par cette régression. Lors d'une classification avec K classes (que l'on peut noter classe $1, \ldots,$ classe K), on aura K neurones dans la couche de sortie. Pour $x \in \mathbb{R}^n$, le neurone $i \in \llbracket 1; K \rrbracket$ on aura une estimation $\tilde{b}_i(x)$, la probabilité que x soit de classe i. La classification s'effectue de telle sorte que $\tilde{cl}(x) \in \arg\max \tilde{b}_i(x)$. On notera M le nombre de neurones dans la couche cachée.

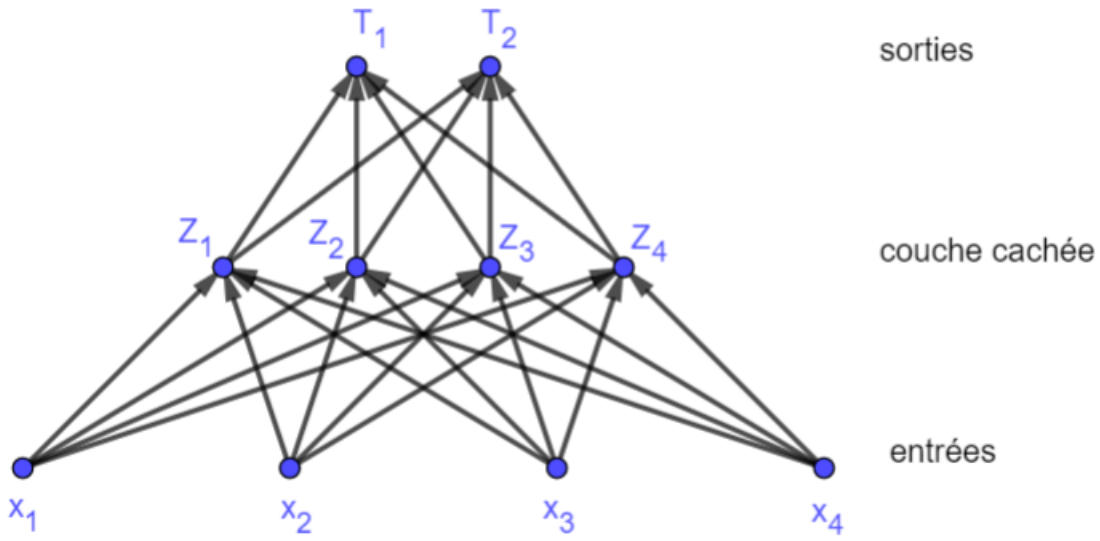


Figure 3.1 Exemple de réseau de neurones à 3 couches en dimension $n = 4$.

Règles de propagation Il faut à présent savoir comment l'information va se transmettre de couche en couche.

$$\forall i \in \llbracket 1; M \rrbracket, Z_i(x) = \sigma(\alpha_i^T x + \alpha_{i0}) \tag{3.2}$$

où σ est appelée «fonction d'activation» et est typiquement $\sigma(v) = \frac{1}{1+\exp(-v)}$ ou une fonction échelon. En notant $Z = (Z_1, \ldots, Z_M)$, on poursuit avec

$$\forall i \in \llbracket 1; K \rrbracket, T_i(Z_1, \ldots, Z_M) = \beta_i^T Z + \beta_{0i}. \tag{3.3}$$

On définit par ailleurs $\forall k \in \llbracket 1; K \rrbracket$, $f_k(x) = g_k(T_1, \ldots, T_K)$ où $g_k(T_1, \ldots, T_K)$ est souvent

l'identité pour la régression (comme dans ce cas on a souvent $K = 1$; alors il s'agit juste de $g_1(T_1) = T_1$). Pour la classification à K classes on choisit souvent :

$$g_k(T_1, \ldots, T_K) = \frac{\exp(T_k)}{\sum_{i=1}^{K} \exp(T_i)}.$$

Apprentissage Les coefficients α_i, α_{i0}, β_i, β_{0i} sont à déterminer lors de la phase d'apprentissage. Une méthode d'apprentissage sur ces coefficients est d'effectuer une rétro-propagation («back-propagation» en anglais) [91].

En notant θ la liste de ces coefficients, pour la régression, on cherche à minimiser

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x^i))^2. \tag{3.4}$$

où les x^i, $i \in [\![1; N]\!]$ sont les éléments de la base de données $\mathcal{C}$ et y_{ik} est la vraie valeur de la k-ème composante analysée par le réseau de neurones pour le point x^i.

Pour la classification, on va vouloir minimiser

$$R(\theta) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \ln(f_k(x^i)). \tag{3.5}$$

Les réseaux de neurones sont assez sensibles au bruit [92] car les réseaux de neurones vont avoir tendance à apprendre le bruit [91] ce qui peut créer des difficultés en optimisation de boites noires de par la nature des problèmes.

3.2.5 Machine à vecteurs de support (SVM)

L'algorithme SVM [93] est un outil de classification qui cherche un hyperplan séparant au mieux les données. Il a connu des transformations pour s'adapter au cas où l'on cherche un classifieur non-linéaire en transférant les données dans un espace où l'on pourra effectuer notre classification linéaire. Nous considérons, pour simplifier les expressions, que les deux classes sont -1 et 1 (au lieu de 0 et 1).

Hyperplan et marge On cherche un hyperplan $\mathcal{H}$, défini par un vecteur $w \in \mathbb{R}^n$ et $w_0 \in \mathbb{R}$, qui sépare le plus clairement possible les données. Plus un élément est éloigné de $\mathcal{H}$, plus il semble qu'il soit bien classé. On souhaite donc que l'ensemble des observations soit le plus loin possible de cet hyperplan.

Définition 3.1. *On définit la marge des observations par rapport à l'hyperplan $\mathcal{H}$ comme $M = \min\{d(v, \mathcal{H}),\ v \in \mathcal{C}\}$ i.e. la plus petite distance entre l'hyperplan et les observations.*

Une fois cette définition posée, l'objectif est de déterminer un hyperplan qui maximise la marge. Un tel hyperplan existe lorsque les données sont linéairement séparables. Dans ce cas, il existe un lien entre la marge et $||w||$ [94], ce qui permet d'obtenir la formulation suivante :

$$\begin{cases} \min\limits_{w \in \mathbb{R}^n} ||w||^2 \\ \text{sous contraintes } b(x^k)(w \cdot x^k + w_0) \geq 1,\ \forall k \in [\![1, N]\!] \end{cases}$$

Dans le cas où les données ne sont pas linéairement séparables [95], un tel hyperplan n'existe pas. Il faut donc une reformulation du problème qui tolère les erreurs tout en les pénalisant. On parle alors d'une marge souple. Cette nouvelle formulation se présente comme suit :

$$\begin{cases} \min\limits_{w,\zeta} \frac{1}{2}||w||^2 + C \sum\limits_{k=1}^{N} \zeta_k \\ \text{sous contraintes } b(x^k)(w \cdot x^k + w_0) \geq 1 - \zeta_k,\ \forall k \in [\![1, N]\!] \\ \qquad\qquad\qquad\qquad \zeta_k \geq 0,\ \forall k \in [\![1, N]\!] \end{cases}$$

avec C un réel strictement positif choisi par l'utilisateur. La valeur de C est en lien avec la tolérance aux points mal-classés [96, 97]. Plus C est grand, plus les points mal-classés sont pénalisés. Les termes ζ_k représentent la pénalisation des points qui ne sont pas mal-classés.

Puis, une dernière modification à l'aide des conditions KKT donne la formulation [98] :

$$\begin{cases} \max\limits_{\alpha \ \mathbb{R}^N} \sum\limits_{i=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i=1}^{N} \sum\limits_{j=1}^{N} b(x^i) b(x^j) \alpha_i \alpha_j {x^i}^T x^j \\ \text{sous contraintes } 0 \leq \alpha_i \leq C, \forall i \in \llbracket 1, N \rrbracket \\ \sum\limits_{i=1}^{N} \alpha_i b(x^i) = 0. \end{cases}$$

Des algorithmes existent pour résoudre cette dernière formulation, par exemple l'algorithme SMO («Sequential Minimal Optimization») [99].

Cas non-linéaire On peut avoir besoin de faire de la classification non-linéaire. Dans ce but, on va placer les éléments de $\mathcal{C}$ dans un espace de dimension supérieure, puis effectuer une classification linéaire à l'aide de l'algorithme SVM avec une méthode identique à celle du cas linéaire [72, 95]. Ces éléments vont être envoyés dans un espace de dimension supérieure à l'aide d'une fonction $\Phi : \mathbb{R}^n \mapsto \mathbb{R}$ continue. Le problème devient alors :

$$\begin{cases} \max\limits_{\alpha \ \mathbb{R}^n} \sum\limits_{k=1}^{N} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{N} b(x^i) b(x^j) \alpha_i \alpha_j \Phi(x^i)^T \Phi(x^j) \\ \text{sous contrainte } 0 \leq \alpha_k \leq C, \forall k \in \llbracket 1, N \rrbracket \\ \sum\limits_{k=1}^{N} \alpha_k b(x^k) = 0. \end{cases}$$

Plus que de dépendre de Φ, le problème dépend de $K(x^i, x^j) = \Phi(x^i)\Phi(x^j)$. La théorie a fait intervenir la fonction Φ, mais, en pratique, K est plus souvent décrit explicitement que Φ [100].

Toute fonction K telle qu'il existe Φ continue telle que $K(x^i, x^j) = \Phi(x^i)\Phi(x^j)$ convient. Pour garantir cela, si un noyau K vérifie le théorème de Mercer, alors K peut s'écrire comme le produit scalaire d'une fonction Φ [101]. On parle dans ce cas d'un noyau de Mercer. Il est à noter que le cas non-linéaire généralise le cas linéaire en prenant Φ comme la fonction identité de $\mathbb{R}^n$, ce qui revient à prendre K comme le produit scalaire canonique de $\mathbb{R}^n$

Cependant, le choix d'un noyau peut créer des problèmes de surapprentissage [102]. Avoir

des informations sur notre problème peut aider à choisir le bon noyau, car sa performance dépend grandement du problème d'optimisation [96].

L'hyperplan, et ses caractéristiques, a été utilisé pour pouvoir effectuer des régressions. Ces régressions sont appelées «Machines à Régression de Vecteurs de Support». Ces régressions sont faites à l'aide de fonctions polynomiales [103].

3.2.6 Forêts aléatoires

Les forêts aléatoires sont un outil de classification qui construit sa décision sur une moyenne de plusieurs arbres de décision. Une forêt aléatoire est constituée de plusieurs arbres qui ont été générés avec une composante aléatoire. Chaque arbre contient un nœud, appelé racine, qui contient tout $\mathbb{R}^n$. Ce nœud racine est ensuite divisé en plusieurs nœuds. Chacun de ces nœuds contient un rectangle de $\mathbb{R}^n$ et l'ensemble de ces nœuds forment une partition de $\mathbb{R}^n$. Récursivement, un nœud, qui représente un rectangle de $\mathbb{R}^n$, peut être divisé, ou non, en deux autres nœuds formant une partition de ce rectangle [104]. Ce partitionnement continue à chaque nœud jusqu'à ce qu'un critère d'arrêt soit atteint. Ces critères peuvent être, par exemple, que l'un des nœuds ne contiendrait pas assez d'éléments de $\mathcal{C}$ ou que l'arbre dépasserait une certaine profondeur [105].

Pour une forêt aléatoire, chaque arbre est construit à l'aide d'une variable aléatoire Z. Les tirages aléatoires sont indépendants et identiquement distribués sur Z [106,107]. Deux arbres différents vont avoir une partition de l'espace différente. Un élément $x^0 \in \mathbb{R}^n$, que l'on souhaite classer, peut se retrouver dans un rectangle très différent d'un arbre à l'autre [108].

Soit $\mathcal{F}$ une forêt aléatoire. On notera $m \in \mathbb{N}$ le nombre d'arbres utilisés dans $\mathcal{F}$. Les arbres seront notés A_i, $i \in \llbracket 1; m \rrbracket$.

Soit $x^0 \in \mathbb{R}^n$, un élément dont on veut prédire la classe. Pour tout $i \in \llbracket 1; m \rrbracket$, on calcule la proportion d'éléments de classe 1 dans la feuille où se trouve x^0 dans l'arbre A_i [104] :

$$\tilde{b^i}(x^0) = \frac{1}{N(A_i(x^0))} \sum_{x \ A_i(x^0)} b(x).$$

Dans cette formule, $A_i(x^0)$ représente le nœud qui contient x^0 dans A_i, et $N(A_i(x^0))$ est le nombre d'éléments de $\mathcal{C}$ contenus dans le rectangle représenté par le nœud $A_i(x^0)$.

Après que la proportion d'éléments de classe 1 est calculée pour chaque arbre A_i, une moyenne est calculée sur tout les arbres de la forêt $\mathcal{F}$ [104].

$$\tilde{b}(x^0) = \frac{1}{m} \sum_{i=1}^{m} \tilde{b}^i(x^0).$$

Il faut noter que, par construction, $\tilde{b}(x^0) \in [0; 1]$ et peut correspondre à ce qui est souhaité dans cette thèse.

Les arbres de forêts aléatoires ont l'avantage de ne pas reposer sur un modèle linéaire ou quadratique. Cependant, cette méthode peut être inefficace lorsque $\mathcal{C}$ ne contient que quelques dizaines d'éléments [105]. Ceci est pourtant le cas en optimisation de boites noires. En effet, $\mathcal{C}$ se construit, à partir d'une liste vide, au fur et à mesure que les points sont évalués par la boite noire.

3.3 Discussion

Après avoir étudié ces algorithmes, l'un d'entre eux sera choisi pour être utilisé dans le reste de la thèse. Cet algorithme doit correspondre aux particularités des problèmes d'optimisation de boites noires.

En optimisation de boites noires, il n'y a pas d'expression analytique des contraintes. Il n'y a pas non plus de connaissance sur le fait que les contraintes soient linéaires, quadratiques, ou autres. Or, la régression logistique, l'analyse discriminante et SVM sont construits sur des modèles linéaires. Ces deux derniers proposent d'utiliser un noyau pour transférer les données dans un espace de plus grande dimension où les données seraient linéairement séparables. Mais quel noyau utiliser quand aucune réelle connaissance des contraintes n'est accessible ?

Les réseaux de neurones ont une mauvaise performance sur les données bruitées car il peut y avoir du surapprentissage sur le bruit.

k-nn ne repose sur aucun modèle. De plus, cet algorithme est plutôt robuste aux données bruitées [74].

En optimisation de boites noires, il y a peu d'éléments dans la base de données, comme nous l'avons rappelé au début du chapitre 2, en particulier en début de résolution d'un problème de boites noires, puisque la cache $\mathcal{C}$ est vide au début et contient des éléments supplémentaires au fur et à mesure que ces éléments ont été évalués. Or, les forêts aléatoires ont besoin d'un ensemble d'entrainement important pour que la phase d'apprentissage soit performante.

L'algorithme k-nn sera préféré aux autres dans les chapitres 4 et 6 car il semble mieux correspondre aux caractéristiques de l'optimisation de boites noires.

CHAPITRE 4 STRATÉGIES D'ORDONNANCEMENT

La stratégie opportuniste est partie prenante de NOMAD, le développement informatique de MADS, qui l'utilise par défaut. La stratégie opportuniste est décrite dans la section 2.3.1 et son intérêt a été montré dans [6]. Or, les différentes stratégies d'ordonnancement dans NOMAD n'ont pas encore été évoquées. L'utilisateur a le choix entre plusieurs possibilités. Il y a l'ordonnancement lexicographique (ordonnancement selon la première coordonnée, puis selon la deuxième, etc) ou selon la direction du dernier succès. Il peut utiliser des modèles ou des fonctions substituts pour la fonction objectif f et/ou pour la fonction de violations des contraintes h, comme un modèle quadratique comme dans [8] ou bien avec les fonctions substituts proposées par [109]. Il peut même définir son propre critère. L'ordonnancement lexicographique donne priorité d'abord aux points avec la plus faible valeur de la première coordonnée. Ceci favorise de façon arbitraire une région de l'espace par rapport à d'autres. Ce critère est donc désactivé par défaut dans NOMAD. En présence de fonctions substituts, il s'agit d'informations utiles et donc l'ordonnancement utilisant les modèles de f et h est prioritaire par défaut. Cependant, dans de nombreux problèmes, aucun modèle n'est disponible. La stratégie d'ordonnancement est alors une stratégie basée sur l'angle fait avec la direction de dernier succès [51, 110].

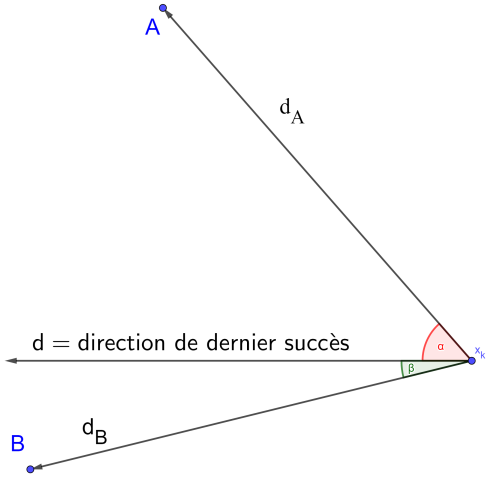


Figure 4.1 Choix d'un point selon la direction de dernier succès.

Cette direction de dernier succès semble prometteuse. Si un succès a été obtenu dans cette direction, d'autres succès peuvent se trouver dans le prolongement. Bien qu'il n'y ait pas de

modèle utilisé, cette direction donne un peu d'information sur la fonction f. La figure 4.1 montre deux points candidats, par rapport à l'itéré courant x_k. A décrit par rapport à x_k la direction d_A et B décrit par rapport à x_k la direction d_B. La direction ayant mené au dernier succès est notée d. α est l'angle géométrique entre d_A et d, β est l'angle géométrique entre d_B et d. Comme $\beta < \alpha$ alors d_B représente une direction plus proche de d que d_A. Ainsi, le candidat B sera placé avant A dans l'ordonnancement. Le critère peut être pris aussi en évaluant B avant A si $\cos(\beta) > \cos(\alpha)$, ou encore $\frac{d_B^T d}{||d_B||||d||} > \frac{d_A^T d}{||d_A||||d||}$.

Ce chapitre propose deux nouvelles stratégies d'ordonnancement. Une première est basée sur la distance des éléments candidats par rapport à la cache. La seconde favorisera les éléments qui seront estimés comme ayant le plus de chances d'être réalisables.

4.1 Ordonnancement selon la distance à la cache

Nous proposons une autre stratégie très simple d'ordonnancement. Il s'agit d'évaluer les points les plus éloignés de ceux qui ont été rencontrés précédemment par l'algorithme. La motivation derrière cette stratégie est de partir vers des régions moins explorées dans l'espoir de trouver des améliorations.

Soit $k \in \mathbb{N}$ le numéro de l'itération. Comme il y a une notion de distance, il faut faire face aux problèmes de mise à l'échelle. Pour cela, la mise à l'échelle dynamique évoquée à la section 2.3.4 sera utilisée. Si à l'itération $k \in \mathbb{N}$, la cache, c'est-à-dire l'ensemble des points déjà évalués par l'algorithme, est notée $\mathcal{C}_k$, et que MADS fournit une liste de points à évaluer $x \in \mathcal{L}_k$, alors le carré de la distance d'un élément $x \in \mathcal{L}_k$ à la cache $\mathcal{C}_k$, en tenant compte de la mise à l'échelle, est

$$d^2(x, \mathcal{C}_k) = \min_{y \ C_k} \sum_{i=1}^{n} \left(\frac{x_i - y_i}{\delta_i^k} \right)^2 .$$

Ainsi cette stratégie d'ordonnancement s'effectue comme à l'algorithme 7.

Résultats numériques On souhaite comparer cet ordonnancement à l'aide de distance à la cache avec celui utilisant la direction de dernier succès. La comparaison va se faire à l'aide de profils de données et de performance, tels que cela a été expliqué à la section 2.4. Les profils vont être effectués sur deux séries de tests. Tous ces tests sont effectués sur NOMAD 3.8.0 avec des directions générées par ORTHOMADS ($2n$ directions [60]) et les modèles quadratiques ont été désactivés.

Algorithme 7 : Ordonnancement selon la distance à la cache

Entrées : $k \in \mathbb{N}$, δ_i^k pour $i \in \llbracket 1; n \rrbracket$, $\mathcal{L}_k \subset \mathbb{R}^n$, la cache $\mathcal{C}_k$.

1 : Ordonnancement des éléments de $\mathcal{L}_k$

Pour chaque $x \in \mathcal{L}_k$.

Calculer $d^2(x, \mathcal{C}_k) = \min_{y \ \mathcal{C}_k} \sum_{i=1}^{n} \left(\frac{x_i - y_i}{\delta_i^k} \right)^2$.

Trier les éléments $x \in \mathcal{L}_k$ par ordre décroissant des valeurs $d^2(x, \mathcal{C}_k)$.

2 : Évaluation opportuniste

Faire évaluer par la boite noire les éléments de $\mathcal{L}_k$ avec la stratégie opportuniste.

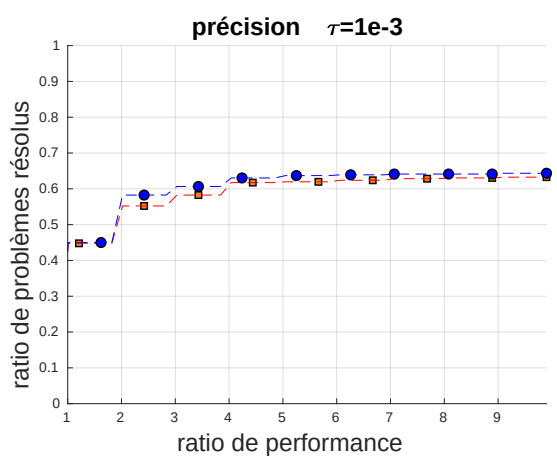
La première série de tests est effectuée sur des problèmes analytiques. Il y a 46 problèmes analytiques différents qui sont tous lancés avec chacun un point de départ réalisable. Ils sont listés dans le tableau 4.1 avec leurs sources, leurs nombres de variables n, leurs nombres de contraintes m et si les variables contiennent des bornes ou non. Pour chacun de ces problèmes, 10 graines aléatoires sont utilisées pour avoir des directions différentes. En effet, les directions D_k comprennent une composante aléatoire. Choisir ces graines aléatoires permet la reproductibilité des résultats. Les résultats numériques sont effectués au total sur 460 différentes instances.

Les graphiques de la figure 4.2 montrent qu'il n'y a pas de différences majeures entre les deux stratégies sur des problèmes analytiques. Il faut donc étudier si les résultats diffèrent sur de vraies boites noires, issues de problèmes industriels.

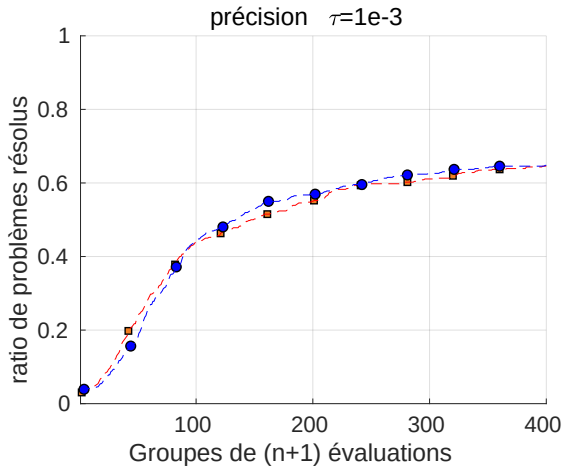
La deuxième série de tests regroupe donc 3 boites noires issues de problèmes industriels. Ces problèmes sont MDO [116], STYRENE [28] et Lockwood [38].

MDO est un problème de portance d'avion avec 10 variables, toutes bornées par 0 et 100, et contenant 10 contraintes.

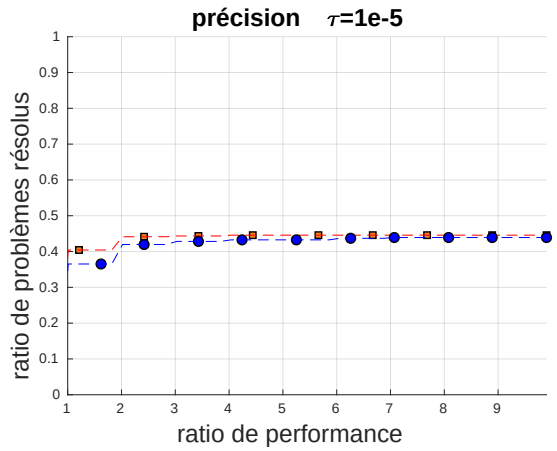
STYRENE est un problème issu de la chimie dans lequel l'objectif est de maximiser la valeur actuelle nette dans la cadre de la production du STYRENE. Ce problème contient 11



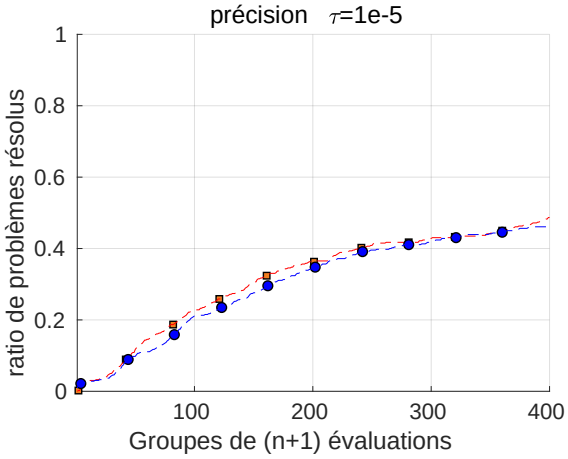
a) Profil de performance $\tau = 10^{-3}$.



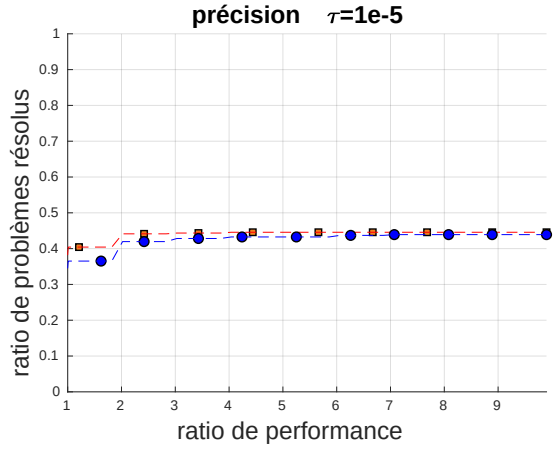
b) Profil de données $\tau = 10^{-3}$.



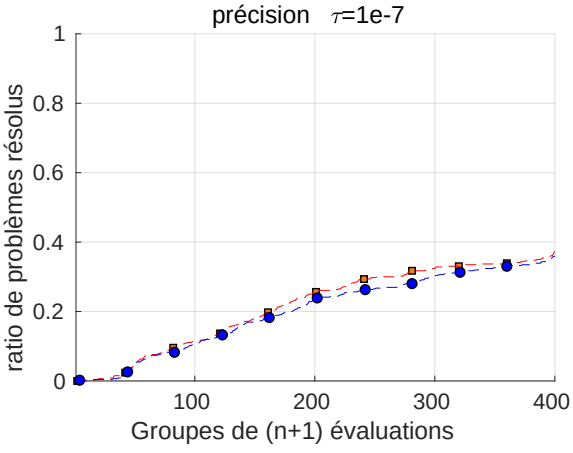
c) Profil de performance $\tau = 10^{-5}$.



d) Profil de données $\tau = 10^{-5}$.



e) Profil de performance $\tau = 10^{-7}$.



f) Profil de données $\tau = 10^{-7}$.

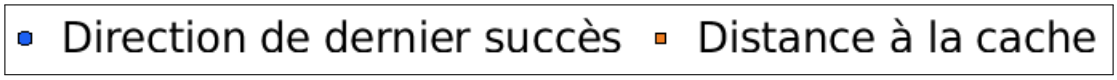


Figure 4.2 Profils de performance et de données avec $\tau = 10^{-3}$, $\tau = 10^{-5}$ et $\tau = 10^{-7}$ pour des problèmes analytiques.

Tableau 4.1 Description de 46 problèmes analytiques de la littérature.

#	Nom	Source	n	m	Bornes
1	ARWHEAD10	[111]	10	0	non
2	ARWHEAD20	[111]	20	0	non
3	BDQRTIC10	[111]	10	0	non
4	BDQRTIC20	[111]	20	0	non
5	BIGGS	[111]	6	0	non
6	BRANIN	[112]	2	0	oui
9	BROWNAL10	[111]	10	0	non
10	BROWNAL20	[111]	20	0	non
11	CRESCENT	[3]	10	2	non
12	DISK	[3]	10	1	non
13	ELATTAR	[113]	6	0	non
14	EVD61	[113]	6	0	non
15	FILTER	[113]	9	0	non
16	G210	[114]	10	2	oui
17	G220	[114]	20	2	oui
18	GOFFIN	[113]	50	1	oui
19	HS78	[115]	5	1	oui
20	HS114	[115]	9	7	oui
21	GRIEWANK	[112]	10	0	oui
22	L1HILB	[113]	50	1	oui
23	MAD6	[113]	5	7	non
24	OSBORNE2	[113]	11	0	non
25	PBC1	[113]	5	0	non
26	PENALTY1_10	[111]	10	0	non
27	PENALTY1_20	[111]	20	0	non
28	PENALTY2_10	[111]	10	0	non
29	PENALTY2_20	[111]	20	0	non
30	PENTAGON	[113]	6	15	non
31	POLAK2	[113]	10	0	non
32	POWELLSG12	[111]	12	0	non
33	POWELLSG20	[111]	20	0	non
34	RASTRIGIN	[112]	2	0	oui
35	SHOR	[113]	5	0	non
36	SNAKE	[3]	2	2	non
37	SROSENBR10	[111]	10	0	non
38	SROSENBR20	[111]	20	0	non
39	TRIDIA10	[111]	10	0	non
40	TRIDIA20	[111]	20	0	non
41	VARDIM10	[111]	10	0	non
42	VARDIM20	[111]	20	0	non
43	WONG1	[113]	7	0	non
44	WONG2	[113]	10	0	non
45	WOODS12	[111]	12	0	non
46	WOODS20	[111]	20	0	non

contraintes, dont 4 qui renvoient des valeurs binaires. Il contient 8 variables.

Lockwood est une version simplifiée d'un problème de pompage et de traitement pour l'assainissement d'eaux souterraines dans le Montana. Il contient 6 variables, et il s'agit d'une version avec 4 contraintes.

D'autres descriptions succinctes de ces problèmes peuvent être trouvées dans les articles [14, 28, 110]. Pour chacun de ces problèmes, 30 points de départ réalisables ont été sélectionnés à l'aide d'un hypercube latin. Les résultats numériques sont donc obtenus au total sur 90 instances différentes.

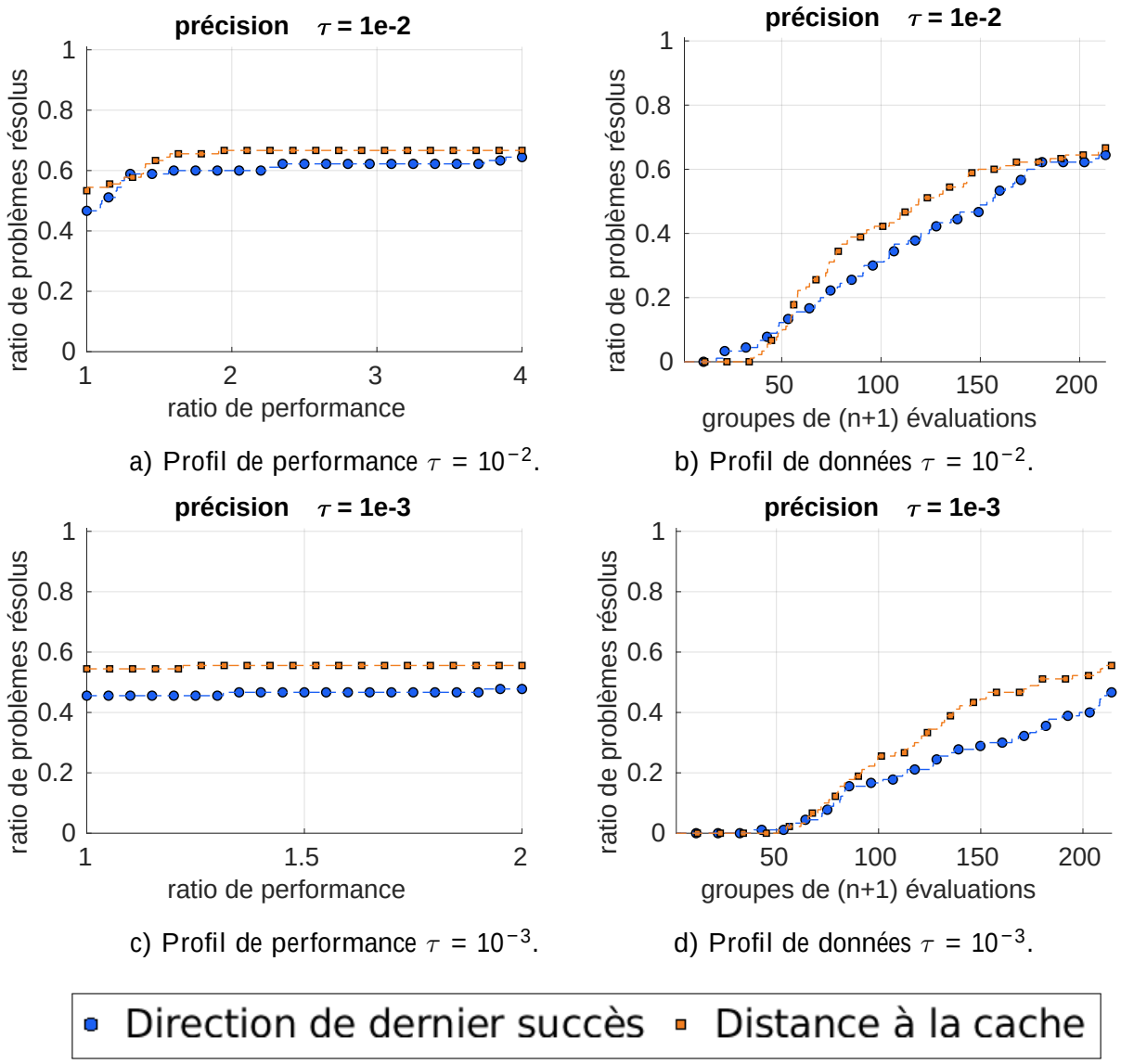
Ce que l'on observe sur les deux graphiques en haut sur la figure 4.3, c'est, qu'à la précision $\tau = 10^{-2}$, la stratégie d'ordonnancement selon la distance à la cache, avec les carrés, domine l'ordonnancement selon la direction de dernier succès, avec les cercles, à la fois sur les profils de données et de performance. Sur le profil de performance, il y a un resserrement vers le ratio de performance $\alpha = 1.3$, voire même la direction de dernier succès passe au dessus. Puis la distance à la cache repasse au-dessus jusqu'à la fin. L'écart le plus important est observé pour $\alpha = 2$ avec un écart de 6.6 points de pourcentage de problèmes résolus. Sur le profil de données, on observe le même type de tendance. Au début, on constate une très légère domination de la direction de dernier succès, mais ensuite la stratégie de distance à la cache la dépasse et la domine. On observe un écart de 15 points de pourcentage à $150 \times (n + 1)$ évaluations.

Cette tendance s'accentue en augmentant la précision à $\tau = 10^{-3}$, sur les deux courbes du bas de la figure 4.3. On observe notamment un écart de 18 points de pourcentage à $180 \times (n+1)$ évaluations sur le profil de données, tandis que sur le profil de performance, on observe un écart de 8 points de pourcentage à $\alpha = 2$.

4.2 Ordonnancement selon la potentielle réalisabilité

Dans cette section, une autre stratégie d'ordonnancement est étudiée. On souhaite ordonner les éléments, de sorte que ceux qui ont le plus de chance d'être réalisables soient évalués en premier. Pour cela, une fonction mesurant ce potentiel de réalisabilité est construite. Il faut que cette fonction ne prenne que des valeurs entre 0 et 1 pour pouvoir l'interpréter comme une estimation de la probabilité d'être réalisable ou non. Or ceci n'est pas le cas pour une régression linéaire ou quadratique, ou pour les interpolations. Pour cela, il est possible d'utiliser des méthodes de régression issues de la classification supervisée pour ordonner une liste de points à évaluer du plus prometteur au moins prometteur. Notons π cette fonction de sorte que pour tout $x \in \mathbb{R}^n$, $\pi(x) \in [0; 1]$.

L'objectif est d'utiliser la méthode k-nn afin de pouvoir créer un ordonnancement d'une liste



a) Profil de performance $\tau = 10^{-2}$.

b) Profil de données $\tau = 10^{-2}$.

c) Profil de performance $\tau = 10^{-3}$.

d) Profil de données $\tau = 10^{-3}$.

Figure 4.3 Profils de performance et de données avec $\tau = 10^{-1}$ et $\tau = 10^{-3}$ sur des boites noires.

$\mathcal{L}_k$ à l'aide d'une fonction, que l'on notera $\pi : \mathbb{R}^n \mapsto [0, 1]$. Plus la valeur $\pi(x)$ est proche de 0, plus la méthode M prédit que x devrait être réalisable. À l'inverse, plus la valeur $\pi(x)$ est proche de 1, plus cette même méthode prédit que x ne devrait pas être réalisable. Ainsi, $\pi(x)$ représente en fait une estimation de la probabilité du point x de ne pas être réalisable d'après k-nn. On supposera que l'on ne dispose pas d'une fonction substitut pour la fonction objectif f. Si l'on se trouve à une itération $k \in \mathbb{N}$ et que MADS fournit une liste $\mathcal{L}_k$ de points à évaluer, l'ordonnancement de $\mathcal{L}_k$ s'effectuera à l'aide de la fonction π.

Une itération de MADS s'effectuera comme dans l'algorithme 8.

Algorithme 8 : Itération de MADS avec un ordonnancement fait à l'aide de k-nn

Entrées : $k \in \mathbb{N}$, $\mathcal{L}_k \subset \mathbb{R}^n$.

1 : Ordonnancement des éléments de $\mathcal{L}_k$

Pour chaque $x \in \mathcal{L}_k$
Calculer $\pi(x)$ à l'aide de k-nn.

Ordonner les éléments $x \in \mathcal{L}_k$ par ordre croissant des valeurs $\pi(x)$.

2 : Évaluation opportuniste

Faire évaluer par la boite noire les éléments de $\mathcal{L}_k$ avec la stratégie opportuniste.

Remarque : Du fait que seul l'ordonnancement des points suggérés est modifié, cette stratégie (comme toute les stratégies d'ordonnancement) n'a pas d'influence sur l'analyse de convergence de l'algorithme utilisé pour résoudre le problème d'optimisation. Elle ne modifie notamment pas celle de MADS évoquée à la section 2.3.3.

Résultats numériques Pour effectuer les tests numériques, les mêmes problèmes qu'à la section 4.1 sont utilisés, à savoir 3 problèmes de boites noires, avec 30 instances pour chacun de ces 3 problèmes, soit donc 90 instances. Ces 3 problèmes sont STYRENE, MDO et Lockwood. Tous ces problèmes contiennent initialement un certain nombre de contraintes et ont été modifiés en remplaçant toutes les contraintes par une unique contrainte binaire indiquant si x est réalisable ou non. Les problèmes sont résolus à l'aide de NOMAD 3.8.0, modifié de façon à proposer une stratégie d'ordonnancement basée sur k-nn. Les directions de sonde seront choisies comme décrites par ORTHOMADS [60]. Les modèles quadratiques sont désac-

tivés. Une instance s'arrête au bout de 1500 évaluations ou si la précision par défaut a été atteinte. Chacun des points de départ de ces instances est réalisable. Aucune fonction substitut ne sera utilisée pour la fonction objectif. La méthode de classification utilisée pour avoir une fonction substitut de la contrainte binaire sera donc la méthode k-nn. On effectuera au préalable des tests préliminaires avec différents types de noyaux. La meilleure méthode sera ensuite choisie pour la comparer avec la méthode par défaut actuelle de NOMAD 3.8.0, en désactivant les modèles, donc avec un ordonnancement basé sur la direction de dernier succès.

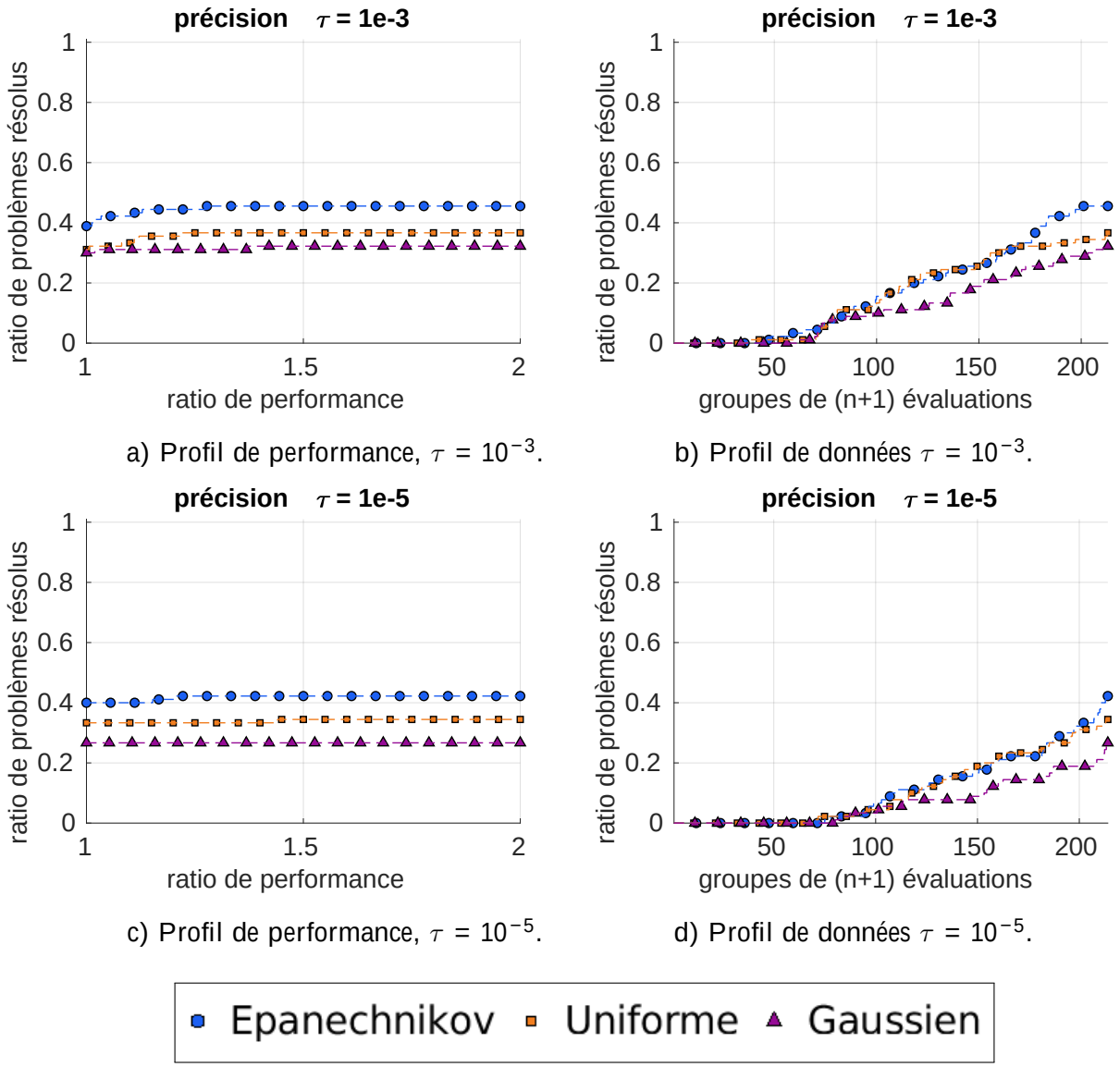
La première méthode utilisera un noyau uniforme avec un nombre de voisins décidé par LOOCV. La deuxième méthode utilisera un noyau d'Epanechnikov avec un nombre de voisins décidé par LOOCV. La troisième utilisera un noyau gaussien avec tous les éléments de la cache comme voisins étant donné qu'il est à support sur tout $\mathbb{R}^n$.

La comparaison va se faire à l'aide de profils de données et de performance, telle qu'expliquée en 2.4.

La figure 4.4 présente les profils de performance et de données aux précisions $\tau = 10^{-3}$ et $\tau = 10^{-5}$ pour les méthodes évoquées ci-dessus. Elle montre que la courbe représentée à l'aide de cercles, qui utilise la stratégie d'ordonnancement utilisant le noyau d'Epanechnikov avec LOOCV, domine globalement les deux autres courbes. C'est donc cette méthode qui sera choisie pour la comparaison avec l'ordonnancement basé sur la direction de dernier succès.

La figure 4.5 présente les profils de performance et de données entre deux stratégies d'ordonnancement. La courbe utilisant les carrés est l'ordonnancement utilisant le noyau d'Epanechnikov avec LOOCV. Celle avec les cercles est l'ordonnancement basé sur la direction de dernier succès. Les deux courbes sont sensiblement les mêmes, et aucune tendance ne se dégage. Une stratégie basée sur la réalisabilité ne présente donc pas d'avantage par rapport à la direction de dernier succès, qui est le critère par défaut de NOMAD dans ce genre de problèmes.

Le manque de risque, dans le fait d'aller tester des régions moins explorées, est lié à l'absence de fonction substitut pour f. Tester d'abord les points qui ont le plus de chances d'être réalisables a tendance à favoriser l'exploration de zones déjà connues. Ceci est mise en évidence sur le problème SNAKE. Il a été définit dans [3] ainsi :



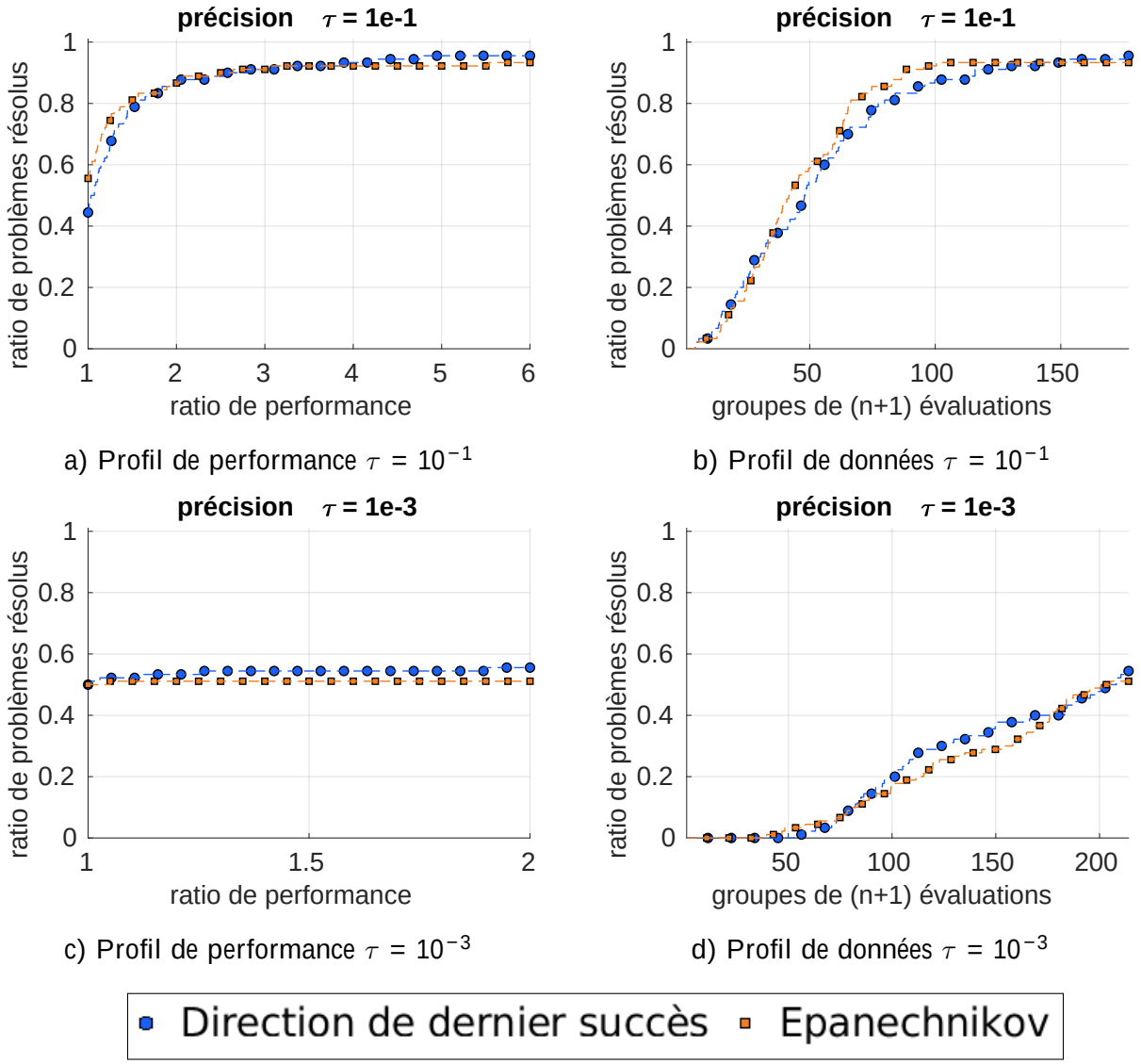
a) Profil de performance, $\tau = 10^{-3}$.

b) Profil de données $\tau = 10^{-3}$.

c) Profil de performance, $\tau = 10^{-5}$.

d) Profil de données $\tau = 10^{-5}$.

Figure 4.4 Profils de performance et de données avec $\tau = 10^{-3}$ et $\tau = 10^{-5}$ sur des boites noires.



a) Profil de performance $\tau = 10^{-1}$

b) Profil de données $\tau = 10^{-1}$

c) Profil de performance $\tau = 10^{-3}$

d) Profil de données $\tau = 10^{-3}$

Figure 4.5 Profils de performance et de données avec $\tau = 10^{-1}$ et $\tau = 10^{-3}$ sur des boites noires.

$$\begin{cases} \min\limits_{x \ \mathbb{R}^2} & \sqrt{(x_1 - 20)^2 + (x_2 - 1)^2} \\ \text{sous contrainte} & \sin(x_1) - \frac{1}{10} \leq x_2 \leq \sin(x_1) \end{cases}$$

Le point de départ réalisable est $(0; 0)^T$. Le point optimal est $x = (20.02887; 0.92434)^T$. Ainsi, ce problème consiste à partir du point de départ à aller à la solution optimale en restant autant que faire se peut entre les deux sinusoïdes.

Or, tous les éléments réalisables sont sur la gauche de la meilleure solution courante. Donc dans le cas où l'on favorise la réalisabilité, les points sur la gauche sont testés d'abord. La figure 4.6 montre les 1000 points évalués avec NOMAD, en haut avec l'ordonnancement de la direction de dernier succès, en bas avec la potentielle réalisabilité basée sur k-nn avec le noyau d'Epanechnikov et LOOCV. Sur la première figure, on observe qu'avec la direction de dernier succès, NOMAD a besoin de peu d'évaluations pour aller d'une crête à une autre. Cela est particulièrement visible dans $[\frac{\pi}{2}; \frac{3\pi}{2}]$ et dans $[\frac{5\pi}{2}; \frac{7\pi}{2}]$ (sur l'axe des abscisses). Dans ce deuxième intervalle, la méthode basée sur la potentielle réalisabilité est très freinée et a besoin de beaucoup d'évaluations pour rejoindre la crête.

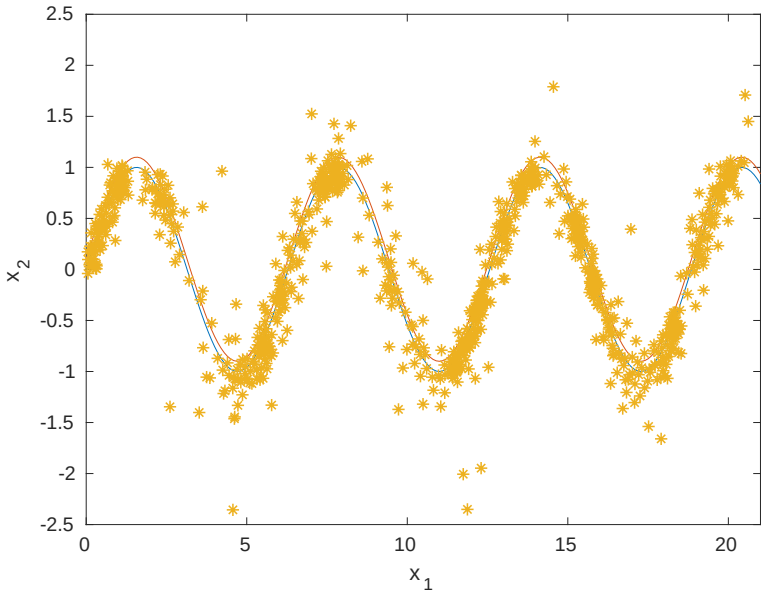
Ainsi, dans le chapitre 6, on proposera d'utiliser, à l'aide des mêmes outils, des fonctions substituts des contraintes binaires. Ainsi, on pourra profiter à la fois de l'information se basant sur la potentielle réalisabilité et de l'information sur la fonction objectif, qui sera plus risquée, dans le but que ces deux informations se complètent.

4.3 Discussion

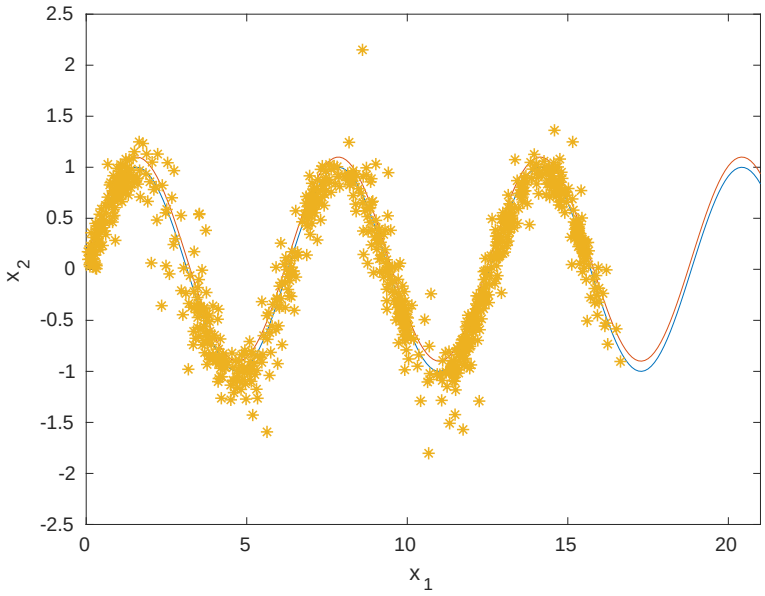
Dans ce chapitre, deux stratégies d'ordonnancement ont été proposées. Elle avaient pour pour but d'être mises en concurrence avec la stratégie basée sur la direction de dernier succès. Cette dernière est la méthode par défaut dans NOMAD lorsque des modèles de f et des contraintes ne sont pas disponibles.

La première méthode, basée sur une stratégie agressive visant à évaluer en priorité les points les plus éloignés de la cache, a montré une amélioration sur les boites noires, sans pour autant en apporter sur des problèmes analytiques.

La deuxième méthode, cherchant en priorité à trouver des points réalisables, n'a montré au-



a) Direction de dernier succès



b) Potentielle réalisabilité

Figure 4.6 En haut, les 1000 points évalués avec comme ordonnancement la direction de dernier succès sur SNAKE. En bas, les 1000 points évalués en favorisant les points qui ont le plus de chances d'être réalisables.

cune amélioration. Ceci semble pouvoir s'expliquer par le côté trop sécuritaire.

Cependant, les outils de cette dernière stratégie pourront peut-être utilisés à nouveau pour faire des modèles de contraintes binaires (et ainsi participer à la création d'une fonction substitut pour h) en combinant cette recherche de points réalisables avec un modèle de f cette fois-ci accessible.

CHAPITRE 5 MISE À L'ÉCHELLE DES SORTIES DANS MADS

Ce chapitre s'intéresse aux problèmes de mise à l'échelle sur les contraintes dans l'algorithme MADS, principalement l'impact sur la fonction de violation des contraintes.

Soient $n, m \in \mathbb{N}$, $X \subset \mathbb{R}^n$, $l, u \in \mathbb{R}^n$. Soit $f : \mathbb{R}^n \mapsto \mathbb{R}$ et pour tout $i \in \llbracket 1, m \rrbracket$, $c_i : \mathbb{R}^n \mapsto \mathbb{R}$ Considérons un problème d'optimisation de boite noire :

$$\begin{cases} \min\limits_{x \ X} f(x) \\ \text{sous contraintes } c_i(x) \leq 0, \ \forall i \in \llbracket 1, m \rrbracket \\ \qquad\qquad l \leq x \leq u. \end{cases}$$

X est l'ensemble qui contient toutes les contraintes non- relaxables. Les $c_i \leq 0$ sont toutes des contraintes relaxables. Il s'agit de la même formulation qu'à la section 2.3.2. $\Omega = \{x \in X : l \leq x \leq u \text{ et } \forall i \in \llbracket 1, m \rrbracket, c_i(x) \leq 0\}$ est le domaine réalisable.

L'algorithme MADS a été étendu [7] afin de prendre en compte différentes échelles en entrée via les pas du treillis selon chaque direction. En revanche, rien n'a été proposé à ce jour pour les contraintes. Il peut être facile de considérer qu'une contrainte puisse retourner en sortie des valeurs positives qui sont souvent très grandes (comme un coût de production en dollars) et d'autre des valeurs positives qui seront en général plus petites (temps d'exécution d'une tâche simple en minutes). Même si, pour ces exemples, l'utilisateur peut avoir conscience de ces ordres de grandeur et rectifier les contraintes en fonction de cela, ces différences d'échelle peuvent créer arbitrairement un déséquilibre entre les contraintes. Cela peut donner plus de poids à quelques contraintes par rapport à d'autres. Une première considération est de remarquer que le problème

$$\begin{cases} \min\limits_{x \ X} f(x) \\ \text{sous contrainte } \frac{c_i(x)}{a_i} \leq 0, \ \forall i \in \llbracket 1; m \rrbracket \\ \qquad\qquad l \leq x \leq u \end{cases}$$

avec, pour tout $i \in \llbracket 1; m \rrbracket$, $a_i \in \mathbb{R}_+$ est équivalent au problème précédent. Or, pour tout $x \in X$, dans le premier cas la fonction de violation de contraintes en x s'écrit $h(x) =$

$\sum_{i=1}^{m} \max(0, c_i(x))^2$ alors que dans le deuxième cas $h(x) = \sum_{i=1}^{m} \max\left(0, \frac{c_i(x)}{a_i}\right)^2$, et les deux problèmes vont être traités différemment par MADS, même si ce sont de par leurs définitions des problèmes strictement équivalents.

Dans tout ce chapitre, la cache à l'itération $k \in \mathbb{N}$ sera notée $\mathcal{C}^k$.

5.1 Pondérations des contraintes relaxables pour h

L'objectif visé par ce travail est de faire une mise à l'échelle des contraintes de sorte qu'une contrainte ne soit pas arbitrairement favorisée par rapport à une autre à cause d'un problème d'échelle, ceci pour que deux problèmes équivalents aient des traitements équivalents. La mise à l'échelle aura lieu pour les contraintes relaxables. Pour les autres, si elles ne sont pas vérifiées, la valeur de h est $+\infty$, donc une mise à l'échelle ne servirait à rien.

Un premier élément à noter, les poids vont changer au fur et à mesure des itérations. Ceci va avoir pour incidence de changer le calcul de la violation des contraintes. Or dans [3], c'est une fonction qui la calcule. Sa définition est donc immuable. Il est donc décidé d'avoir une fonction de violation par itération de l'algorithme de MADS dans le cadre théorique de ce travail. Chaque fonction de violation des contraintes est elle-même dépendante des poids de l'itération courante. Ainsi, pour tout $i \in \llbracket 1; m \rrbracket$, on notera $(a_i^k)_{k \in \mathbb{N}} \in (\mathbb{R}_+)^{\mathbb{N}}$, la suite des poids pour la i-ème contrainte. Ceci permet donc de définir la suite des fonctions de violation des contraintes $(h^k)_{k \in \mathbb{N}}$ par son terme général tel que pour tout $k \in \mathbb{N}$, $h^k : \mathbb{R}^n \mapsto \mathbb{R}_+ \cup \{+\infty\}$ tel que pour tout $x \in \mathbb{R}^n$,

$$h^k(x) = \begin{cases} \sum_{i=1}^{m} \max\left(0, \frac{c_i(x)}{a_i^k}\right)^2 & \text{si } x \in X \\ +\infty & \text{sinon.} \end{cases}$$

Remarque : h^k ne représente pas ici la fonction h à la puissance k. Cette notation est faite pour coïncider avec h^k_{max} de la section 2.3.2.

Ceci a forcément un impact sur la barrière progressive puisqu'elle utilisait à la base un diagramme de h vs f sur lequel à chaque $x \in \mathcal{C}^k$, il y avait un point associé dans ce diagramme qui avait en abscisse $h(x)$ et en ordonnée $f(x)$. Dans le cadre de ce chapitre, chacune des itérations aura son diagramme h^k vs f fonctionnant sur le même principe, la barrière et les

points non-dominés devant théoriquement être recalculés intégralement au début de chaque itération. En pratique, à une itération $k \in \mathbb{N}$, il n'y a pas toujours de changements entre a_i^k et a_i^{k+1}. Si ceci est le cas pour tout $i \in \llbracket 1; m \rrbracket$, alors $h^k = h^{k+1}$. Ainsi, les points dominés au début de l'itération k restent dominés au début de l'itération $k+1$. Il n'est donc pas nécessaire de déterminer à nouveau les points dominés depuis le début.

Si pour un $k \in \mathbb{N}$, $h^k \neq h^{k+1}$, alors l'emplacement des points de la cache va changer entre les diagrammes «h^k vs f» et «h^{k+1} vs f». Ceci amène quelques conséquences concernant le fonctionnement de la barrière progressive.

La figure 5.1 montre un exemple de diagrammes «h^k vs f» au début d'une itération k et $k+1$, $k \in \mathbb{N}$. Dessus, le point A qui était au-delà de la barrière passe de l'autre côté et devient même un point non-dominé. Le point B, qui était non-dominé, reste du bon côté de la barrière mais devient dominé par le point C. Le point D, qui était non-dominé, se retrouve même de l'autre côté de la barrière.

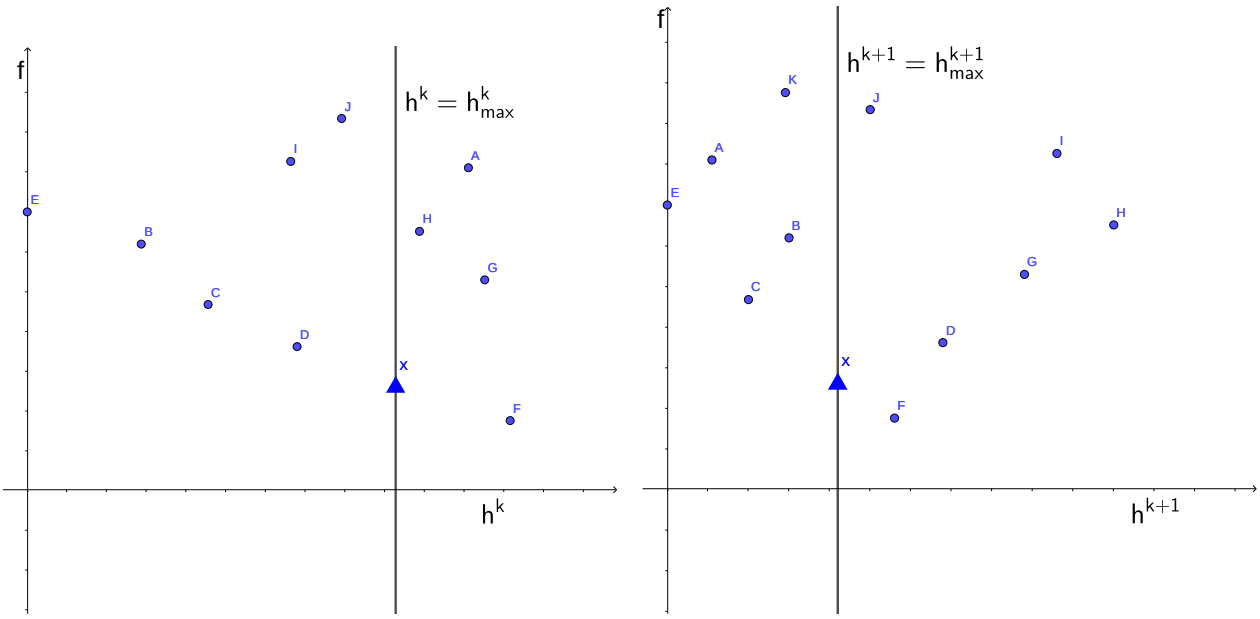
Plusieurs façons de pondérer vont être évoquées. Cependant, il est très important que la propriété 2.2 reste vérifiée. Une version alternative est proposée qui s'adaptera au cadre de ce chapitre.

Propriété 5.1. *Soient $x \in X$ et $(a_i^k)_{k\ \mathbb{N}} \in (\mathbb{R}_+)^{\mathbb{N}}$. On a l'équivalence :*
$h^k(x) = 0$ *si et seulement si x est réalisable par rapport à*
$\Omega = \{x \in X : \forall i \in \llbracket 1, m \rrbracket,\ c_i(x) \leq 0\}$.

5.2 Trois possibilités de pondérations de la violation des contraintes

Trois pondérations différentes sont proposées dans cette section. Les trois pondérations seront initialisées : pour tout $i \in \llbracket 1; m \rrbracket$, $a_i^0 = 1$. Cependant, les deux premières verront les poids ne changer qu'au plus une fois par contrainte, tandis que la troisième nécessitera de changer un nombre indéfini de fois.

Pondération par la première violation La première pondération cherche à effectuer une mise à jour le plus rapidement possible. Pour cela, pour une contrainte donnée, sa pondération sera effectuée par la valeur obtenue lors de la première violation de celle-ci. En d'autres termes, soit $i \in \llbracket 1; m \rrbracket$, alors a_i sera égal à la valeur $c_i(x)$ pour le premier $x \in \mathcal{C}^k$ tel que $c_i(x) > 0$. Ainsi, la pondération a lieu dès qu'elle a de l'intérêt, c'est-à-dire dès que l'on



a) Au début de l'itération k. b) Au début de l'itération k + 1.

Figure 5.1 Mise à jour de h et de h_{max}.

identifie un candidat x qui viole la contrainte : $c_i(x) > 0$. De plus, une fois modifié, a_i^k est fixé pour le reste du déroulement de l'algorithme.

La question du recalcul des $h^k(x)$, $x \in \mathcal{C}^k$ ne se pose qu'une seule fois par contrainte. Néanmoins, deux problèmes peuvent être anticipés. Cette valeur est intégralement dépendante de celle trouvée, qui peut être beaucoup plus grande ou beaucoup plus petite que les valeurs trouvées en général pour cette contrainte. Ceci peut créer un déséquilibre par rapport aux autres contraintes. Rappelons que, pour tout $i \in [\![1, m]\!]$ et tout $k \in \mathbb{N}$, une contrainte binaire a toujours dans ce cas une valeur $a_i^k = 1$ et donc que les $\frac{c_i(x)}{a_i^k}$ valent tous 0 ou 1. L'ordre de grandeur souhaité est de l'ordre de 1, ce que cette façon de calculer a_i^k ne peut garantir pour une contrainte à valeurs continues. L'autre problème est que si la valeur de a_i^k est très faible, des difficultés de calculs liées à des divisions de nombres petits sont à envisager.

Pour cela une deuxième pondération va utiliser la médiane pour atténuer ces défauts.

Pondération par la médiane La deuxième option est d'attendre que la contrainte soit violée un certain nombre de fois, pour avoir un échantillon représentatif. Typiquement, à

l'itération $k \in \mathbb{N}$, pour calculer a_i^k, on regarde si la contrainte c_i est violée n fois, où n est le nombre de variables. Si ce n'est pas le cas, alors $a_i^k = 1$. Sinon, notons $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}^n$ les n premiers points qui violent la i-ème contrainte, triés par ordre croissant de valeurs de $c_i(x_j)$, $j \in \llbracket 1; n \rrbracket$. Alors on pose,

$$a_i^k = c_i(x_{\lceil \frac{n}{2} \rceil}). \tag{5.1}$$

Comme pour la pondération par la première violation, il faut noter que le changement de pondération ne se fera qu'au plus une seule fois par contrainte.

Même si les risques de tomber sur une pondération non-adaptée sont réduits, du fait de prendre la médiane par rapport à la première violation d'une contrainte, il est toujours possible que les n premières violations ne soient pas représentatives non plus des valeurs prises habituellement par cette contrainte.

Pondération par le maximum La troisième pondération introduit pour tout $i \in \llbracket 1; m \rrbracket$ la suite $(a_i^k)_{k \in \mathbb{N}}$ de terme général

$$a_i^k = \begin{cases} 1 & \text{si } \{c_i(x) : x \in \mathcal{C}^k, c_i(x) > 0\} = \emptyset \\ \max\limits_{x \in \mathcal{C}^k} c_i(x) & \text{sinon.} \end{cases}$$

Par construction, pour tout $x \in \mathcal{C}^k \cup X$, $\max\left(0, \frac{c_i(x)}{a_i^k}\right) \in [0; 1]$. De ce fait, les contraintes sont davantage comparables entre elles, y compris avec les contraintes binaires. Il faut noter également que cette mise à l'échelle dynamique s'étend parfaitement aux contraintes binaires puisque ces coefficients pour les mises à l'échelle pour ces contraintes vaudraient 1.

Il faut préciser aussi que lorsque $\{c_i(x) : x \in \mathcal{C}^k, c_i(x) > 0\} \neq \emptyset$, alors cet ensemble est une partie finie de $\mathbb{R}$ et possède dès lors un plus grand élément. Ceci définit correctement a_i^k.

Cette façon de calculer les a_i^k engendre trois problèmes. Le premier est que si l'on trouve une valeur particulièrement grande par rapport aux autres violations typiques de cette même contrainte, a_i^k risque de baisser trop fortement $\frac{c_i(x)}{a_i^k}$, ce qui risque d'abaisser trop fortement l'importance globale accordée à cette contrainte. Le deuxième problème est que dans certaines boites noires qui sont des codes informatiques, lorsqu'il y a une erreur (par exemple provoquée par une contrainte cachée), une valeur arbitrairement élevée (comme 10^{20}) est renvoyée. Dans ce cas, cela a un impact direct sur la valeur a_i^k et sur le calul de h^k. Le troisième

problème est d'être obligé de mettre à jour a_i^k à chaque nouvelle plus grande valeur positive trouvée pour cette contrainte. Ainsi, il faut constamment vérifier s'il ne faut pas remettre à jour les poids, à l'inverse des autres méthodes où le changement de pondération ne s'effectue qu'une seule fois par contrainte.

5.3 Mise à jour de la cache et de h_{max}

Lorsque qu'il n'y a pas de mise à l'échelle des sorties, la fonction h est toujours la même au fur et à mesure des itérations. La fonction de violation des contraintes est calculée par la même formule pour tous les éléments de la cache et n'est jamais recalculée. Ceci n'est plus le cas s'il y a une mise à l'échelle des sorties, étant donné que les pondérations changent la façon de calculer h. Pour pallier cela, il est possible de recalculer h pour tous les éléments de la cache dès que l'on sait que les coefficients de pondération ont changé.

Le fait de recalculer h pour chaque élément de la cache, via la suite $(h^k)_{k\ \mathbb{N}}$, a une conséquence sur l'application de la barrière progressive, notamment s'il n'y a pas de mise à jour de h_{max}^k cohérente avec $(h^k)_{k\ \mathbb{N}}$. Il se peut qu'à une itération $k \in \mathbb{N}$, après calcul, tous les points passent de l'autre côté de la barrière, c'est-à-dire que l'on ait pour tout $x \in \mathcal{C}^k$ non-réalisable, $h^k(x) > h_{max}^k$. Or, dans ce cas, tous les points de la cache sont rejetés par la barrière progressive. Faute de points, NOMAD, l'implémentation de MADS, s'arrête. Pour cela, une mise à jour de h_{max}^k est nécessaire.

La figure 5.1 montre comment choisir le nouveau h_{max}^k.

À une itération $k \in \mathbb{N}$ de MADS avec la barrière progressive, si l'on a au moins un point non-réalisable, alors h_{max}^k est choisi de façon à avoir forcément la même valeur de l'un des $h(x)$ avec un certain $x \in \mathcal{C}^k$. Dans ce projet, on souhaite conserver cette propriété. Ainsi, lorsque la droite d'équation $h^k = h_{max}^k$ passe par l'un des points dans le diagramme h^k vs f de la barrière progressive. Appelons ce point $x_{max}^k \in \mathbb{R}$, représenté par le triangle X dans le diagramme de gauche h^k vs f de la figure 5.1. Alors après mise à jour de h du point x_{max}^k le point X est déplacé sur le diagramme h^{k+1} vs f par rapport au diagramme h^k vs f, comme sur la figure 5.1. C'est le cas pour tous les autres points qui voient leurs valeurs de h modifiées (à noter que les points réalisables, comme le point E, qui sont sur l'axe des abscisses, eux nécessairement ne bougent pas car ils ont toujours la valeur 0 par les fonctions h^k). Lorsque $h^k \neq h^{k+1}$, il est décidé de choisir h_{max}^{k+1} de telle sorte que la droite d'équation $h = h_{max}^{k+1}$ passe

toujours par le point X dans le diagramme h^{k+1} vs f. Ainsi, il y a la garantie qu'au moins un point ne sera pas coupé par la barrière progressive, à savoir le point x^k_{max}, toujours représenté par X. C'est ce que l'on appellera «la mise à jour de h^k_{max}».

5.3.1 Analyse de convergence

Cette mise à l'échelle des sorties est en lien avec plusieurs aspects de la barrière progressive, qui a été présentée en 2.3.2. Or, l'analyse de convergence de MADS est dépendante de la barrière progressive, dans les cas où certaines contraintes l'utilisent. Il convient de vérifier les modifications que cela pourrait avoir dans l'analyse de convergence. Plusieurs pondérations ont été vues et elles n'ont pas toutes la même incidence sur l'analyse de convergence.

Pondération par la première violation et par la médiane Les deux pondérations sont regroupées car elles ont une analyse identique. Tout d'abord, il faut noter que dans les deux cas, il n'y a réellement qu'au plus un seul changement dans les poids pour chaque contrainte, le poids étant de 1 au début et valant ensuite une valeur que l'on notera $a \in \mathbb{R}_+$. Ainsi, pour tout $i \in \llbracket 1; m \rrbracket$, il existe $k_i \in \mathbb{N}$ tel que pour tout $k \in \llbracket 0; k_i \rrbracket$, $a^k_i = 1$ et pour tout $k \in \llbracket k_i; +\infty \llbracket$, $a^k_i = a_i > 0$. Ainsi $(a^k_i)_{k \in \mathbb{N}}$ converge vers une valeur strictement positive. Il en découle par construction de $(h^k)_{k \in \mathbb{N}}$ que cette suite prend à partir du rang $k_{max} = \max_{i \in \llbracket 1; m \rrbracket} k_i$ la valeur h qui est la fonction

$$h(x) = \begin{cases} \sum_{i=1}^{m} \max\left(0, \frac{c_i(x)}{a_i}\right)^2 & \text{si } x \in X \\ +\infty & \text{sinon.} \end{cases}$$

Ainsi, à partir de l'itération k_{max}, la suite $(h^k)_{k \in \mathbb{N}}$ prend toujours la même valeur donc l'analyse de convergence de la barrière progressive évoquée dans 2.3.2 et décrite dans [3] est valide à partir de ce rang.

Pondération par le maximum Le type de raisonnement qui a été utilisé pour l'analyse de convergence avec la pondération par la médiane ne peut plus être utilisé dans le cas d'une pondération par le maximum. En effet, il existe des exemples où l'on peut montrer que l'on convergera vers la solution optimale mais avec une infinité de pondérations si l'on donne un budget d'évaluation infini. Posons $f : [0; 2] \mapsto \mathbb{R}$ définie par son terme général $f(x) = -x$ et $c : [0; 2] \mapsto \mathbb{R}$ défini par son terme général

$$c(x) = \begin{cases} 0 & \text{si } x \in [0;1] \\ 3-x & \text{si } x \in]1;2]. \end{cases}$$

Le problème d'optimisation

$$\begin{cases} \min\limits_{x \in \mathbb{R}} f(x) \\ \text{sous contrainte } c(x) \leq 0 \\ \qquad\qquad\qquad 0 \leq x \leq 2 \end{cases}$$

admet comme solution optimale $x^* = 1$.

Si on lance l'algorithme MADS avec comme point de départ $x_0 = 0.9$, on va générer des points proches de x^* de part et d'autre. Au fur et à mesure, le pas du maillage va diminuer et on va avoir des points plus grands strictement que 1 et de plus en plus proches de 1, donc ayant des valeurs par la contrainte c de plus en plus grandes. Ainsi, on trouvera une infinité de points qui donneront des valeurs de plus en plus grandes par la fonction c, donc qui provoqueront une infinité de mises à jour de h.

Cependant, certains résultats peuvent être prouvés.

Théorème 5.1. *La suite de fonctions* $(h^k)_{k \in \mathbb{N}}$ *converge.*

Démonstration : Pour tout $i \in [\![1; m]\!]$, il faut noter que la suite $(a_i^k)_{k \in \mathbb{N}}$ est croissante à partir d'un certain rang (à partir de la première violation de la i-ème contrainte, si cette contrainte est violée une fois, soit à partir du rang 0) par construction de cette pondération.

Remarque : On précise ici ce qui est sous-entendu par une suite «croissante» (respectivement «décroissante»). Soit $(u_k)_{k \in \mathbb{N}}$ une suite à valeurs réelles. Alors $(u_k)_{k \in \mathbb{N}}$ est croissante (respectivement «décroissante») si pour tout $k \in \mathbb{N}$, $u_k \leq u_{k+1}$ (respectivement $u_k \geq u_{k+1}$).

Comme $(a_i^k)_{k \in \mathbb{N}}$ est croissante à partir d'un certain rang et à valeurs strictement positives, il existe deux cas de figure. Soit cette suite est majorée et elle converge vers une valeur $a_i > 0$, soit elle n'est pas majorée et elle diverge vers $+\infty$. Dans tous les cas, $\left(\frac{1}{a_i^k}\right)_{k \in \mathbb{N}}$ converge vers une valeur positive ou nulle. De la convergence de cette dernière suite, il en découle que la suite $(h^k)_{k \in \mathbb{N}}$ converge. D'où le résultat.

Le théorème suivant s'intéresse à la décroissance de la suite $\{h^k_{max}\}_{k\ \mathbb{N}}$. C'était l'une des propriétés de l'analyse de convergence de la barrière progressive [3].

Théorème 5.2. *La suite $\{h^k_{max}\}_{k\ \mathbb{N}}$ est décroissante à partir d'un certain rang.*

Démonstration : La démonstration du dernier théorème a souligné que, pour tout $i \in \mathtt{J1;}m\mathtt{K}$ la suite $(a^k_i)_{k\ \mathbb{N}}$ est croissante à partir d'un certain rang. Notons k_i ce rang, et notons $k_{max} = \max_{i\ \mathtt{J1;}m\mathtt{K}} k_i$. À partir du rang k_{max}, soit la valeur de h^k_{max} change du fait du changement de la pondération comme expliqué à la section 5.3, soit du fait du fonctionnement de la barrière progressive, comme expliqué dans [3] et dans 2.3.2. Dans le second cas, on sait que h^k_{max} est mis à jour avec une valeur plus petite. Dans le premier cas, soit $k \in \mathbb{N}$, $k \geq k_{max}$ et soit x l'élément qui a servi pour la mise à jour de h^k_{max}. Alors $h^k_{max} = h^k(x)$ et $h^{k+1}_{max} = h^{k+1}(x)$. Il faut donc prouver que $h^{k+1}(x) \leq h^k(x)$. Or comme, $k \geq k_{max}$, alors pour tout $i \in \mathtt{J1;}m\mathtt{K}$, $0 < a^k_i \leq a^{k+1}_i$, donc $0 < \frac{1}{a^{k+1}_i} \leq \frac{1}{a^k_i}$. Ceci étant vrai pour tout $i \in \mathtt{J1;}m\mathtt{K}$, il en découle que $h^{k+1}(x) \leq h^k(x)$. Ainsi, $h^{k+1}_{max} \leq h^k_{max}$. On a montré qu'à partir du rang k_{max}, qu'importe la manière dont la barrière était mise à jour, $h^{k+1}_{max} \leq h^k_{max}$. D'où le résultat.

Une démonstration quasiment identique peut être utilisée pour montrer que, pour tout $x \in X$, la suite $\{h^k(x)\}_{k\ \mathbb{N}}$ est décroissante (au sens large, comme décrit pour les suites) à partir du rang k_{max}. Par ailleurs, par construction, la suite $\{h^k_{max}\}_{k\ \mathbb{N}}$ est minorée par 0. Comme, en plus, elle est décroissante à partir d'un certain rang, alors elle converge.

Il reste une propriété assez proche de la décroissance de $(h^k(x))_{k\ \mathbb{N}}$ pour tout $x \in X$ à partir du rang k_{max}. En fait, les points ne vont jamais voir leurs valeurs diminués par la suite $(h^k(x))_{k\ \mathbb{N}}$ dès qu'ils sont évalués. Ceci est résumé par le théorème 5.3.

Théorème 5.3. *Soit $k_0 \in \mathbb{N}$. Pour tout $x \in \mathcal{C}^{k_0} \cap X$, la suite $\{h^k(x)\}_{k\ \mathtt{Jk_0;+\infty J}}$ est décroissante.*

Démonstration : Soient $k_0 \in \mathbb{N}$ et $A = \{i \in \mathtt{J1;}m\mathtt{K} : \forall x \in \mathcal{C}^{k_0}\ c_i(x) \leq 0\}$. A est l'ensemble des indices des contraintes qui n'ont pas été violées au début de l'itération k_0. Soit $x \in \mathcal{C}^{k_0} \cap X$. Alors, pour tout $k \in \mathtt{J1;}m\mathtt{K}$:

$$\begin{aligned} h^k(x) &= \sum_{i=1}^{m} \max\left(0, \frac{c_i(x)}{a_i^k}\right)^2 \\ &= \sum_{i \,/\, A} \max\left(0, \frac{c_i(x)}{a_i^k}\right)^2 + \sum_{i \;\; A} \max\left(0, \frac{c_i(x)}{a_i^k}\right)^2 \\ &= \sum_{i \,/\, A} \max\left(0, \frac{c_i(x)}{a_i^k}\right)^2 \text{ (par définition de } A) \end{aligned}$$

De plus, pour tout $i \notin A$, il existe $x^i \in \mathcal{C}^{k_0}$, tel que $c_i(x^i) > 0$. Or la suite $\{a_i^k\}_{k \;\mathbb{N}}$ est croissante à partir de la première violation de la contrainte c_i. Donc, pour tout $i \notin A$, $\{a_i^k\}_{k \;]k_0;+\infty[}$ est croissante et à valeurs strictement positives. D'où, pour tout $i \notin A$, $\{\frac{1}{a_i^k}\}_{k \;]k_0;+\infty[}$ est décroissante et à valeurs strictement positives. Il en découle que $\{h^k(x)\}_{k \;]k_0;+\infty[}$ est décroissante. D'où le résultat.

Ce théorème 5.3 montre donc que les seuls éléments $x \in X$ pour lesquels la suite de $\{h^k(x)\}_{k \;]k_0;+\infty[}$ n'est pas décroissante, et donc peut encore prendre parfois des valeurs plus grandes, sont parmi les points qui n'ont pas encore été évalués. Cela montre aussi qu'à partir du diagramme h^{k_0} vs f, un point ne va jamais se déplacer vers la droite dans le diagramme h^{k_0+1} vs f.

La convergence de la suite $\{h^k\}_{k \;\mathbb{N}}$ a été évoqué. Sans information supplémentaire, c'est ce que l'on appelle une convergence simple. Mais en rajoutant une hypothèse sur le fait que les fonctions c_i soient majorées sur X, la convergence uniforme peut être prouvée, qui est plus forte que la convergence simple.

Théorème 5.4. *Si pour tout $i \in]1;m[$, les fonctions c_i sont majorées sur X, alors $\{h_k\}_{k \;\mathbb{N}}$ converge uniformément vers h sur X.*

On prendra la convention suivante pour la démonstration : si $a_i = +\infty$, alors $\frac{1}{a_i} = 0$.
Démonstration :

Soit $x \in X$ et $k \in \mathbb{N}$. Puisque pour tout $i \in]1;m[$, les fonctions c_i majorées sur X, alors $C = \max\limits_{i \;\;]1;m[} \{\sup\{\max(0, c_i(x)), x \in X\}\}$ est correctement défini.

$$\begin{aligned} \left|h^k(x) - h\ (x)\right| &= \left|\sum_{i=1}^{m} \left(\max(0, \frac{c_i(x)}{a_i^k}\right)^2 - \max\left(0, \frac{c_i(x)}{a_i}\right)^2\right| \\ &= \left|\sum_{i=1}^{m} \left(\frac{1}{(a_i^k)^2} - \frac{1}{(a_i)^2}\right) \max(0, c_i(x))^2\right| \\ &\leq C^2 \sum_{i=1}^{m} \left|\frac{1}{(a_i^k)^2} - \frac{1}{(a_i)^2}\right| \end{aligned}$$

Cette dernière inégalité étant vraie pour tout $x \in X$, alors

$$\sup_{x \ X} \left| h^k(x) - h\ (x) \right| \leq C^2 \sum_{i=1}^{m} \left| \frac{1}{(a_i^k)^2} - \frac{1}{(a_i)^2} \right| \longrightarrow 0$$

Ce qui prouve le théorème 5.4.

Points à noter sur l'implémentation pratique de la mise à l'échelle NOMAD 3.8.0 propose une implémentation en C++ de la barrière progressive pour MADS. À ce titre, NOMAD conserve les points de la barrière avec notamment tous les points non-dominés et non-réalisables. Dans cette version par défaut, au début de chaque itération $k \in \mathbb{N}$, la barrière n'est pas recalculée depuis le début. En effet, un point qui n'est pas dans la barrière est un point dominé. Or comme il n'y a pas de pondération, ce point dominé restera dominé pour tout le déroulement de la résolution par NOMAD. Dans NOMAD, la fonction de violation des contraintes est décrite. Ainsi, il n'y a pas besoin de vérifier à nouveau s'il peut faire partie de la barrière. Il en résulte que les seuls points qui peuvent faire partie de la barrière sont ceux qui y étaient au début de l'itération k et ceux qui ont été évalués pendant l'itération k. NOMAD 3.8.0 procède donc ainsi, pour tous les éléments évalués $x \in V_k$, on vérifie si x est dominé par un élément de la barrière, si ce n'est pas le cas alors il doit être rajouté à la barrière et il faut retirer les éléments de la barrière qu'il domine.

Cette stratégie doit être ajustée lorsqu'il y a des pondérations comme proposées dans ce chapitre. En effet, comme les valeurs de h peuvent changer, dans NOMAD, pour tous les points de la cache, alors un point dominé peut devenir non-dominé. Ainsi, à chaque fois qu'il faut recalculer toutes les valeurs de h pour la cache dans NOMAD, il faut déterminer à nouveau la barrière en la recalculant depuis une liste vide.

5.3.2 Résultats numériques

Tous les résultats numériques sont effectués sur NOMAD 3.8.0 avec les directions de ORTHOMADS [60], en désactivant les modèles et un budget de 1500 évaluations.
Les résultats numériques sont répartis sur deux familles de problèmes. La première est composée de problèmes analytiques, qui sont énumérés dans le tableau 5.1. La deuxième famille de problèmes est un ensemble de boites noires, les mêmes qu'en section 4.1. Pour chaque problème «original», une version modifiée, dite «déséquilibrée», a été construite. Ce déséquilibrage a été fait de façon arbitraire sur certaines contraintes en les multipliant chacune par un coefficient 10^{j_i}, $j_i \in \mathbb{Z}$.

Pour chacun des problèmes, des points de départ réalisables ont été générés à l'aide d'un hypercube latin, soit sur tout le domaine, soit autour d'un point réalisable connu.

Pour comparer les différentes versions, des profils de données et de performance, tel que décrits à la section 2.4, sont générés.

Problèmes analytiques Des tests seront d'abord faits sur 5 problèmes analytiques non modifiés. Pour chaque problème, il y aura 100 points de départ réalisables. Ceci fera 500 instances au total. Ensuite, les 5 problèmes seront modifiés pour que les contraintes soient «déséquilibrées». Chaque problème aura les mêmes 100 points de départ réalisables que pour les problèmes non modifiés, soit encore 500 instances.

#	Nom	Source	n	m	Bornes
1	PIGACHE	[117]	4	11	oui
2	PRESSURE VESSEL MIXED CASE1	[118]	4	3	non
3	REINFORCED CONCRETE BEAM MOD1	[119]	3	2	oui
4	SPEED REDUCER MOD MIXED CASE1	[118]	7	1	oui
5	SPRING MOD MIXED CASE1	[120]	3	4	oui

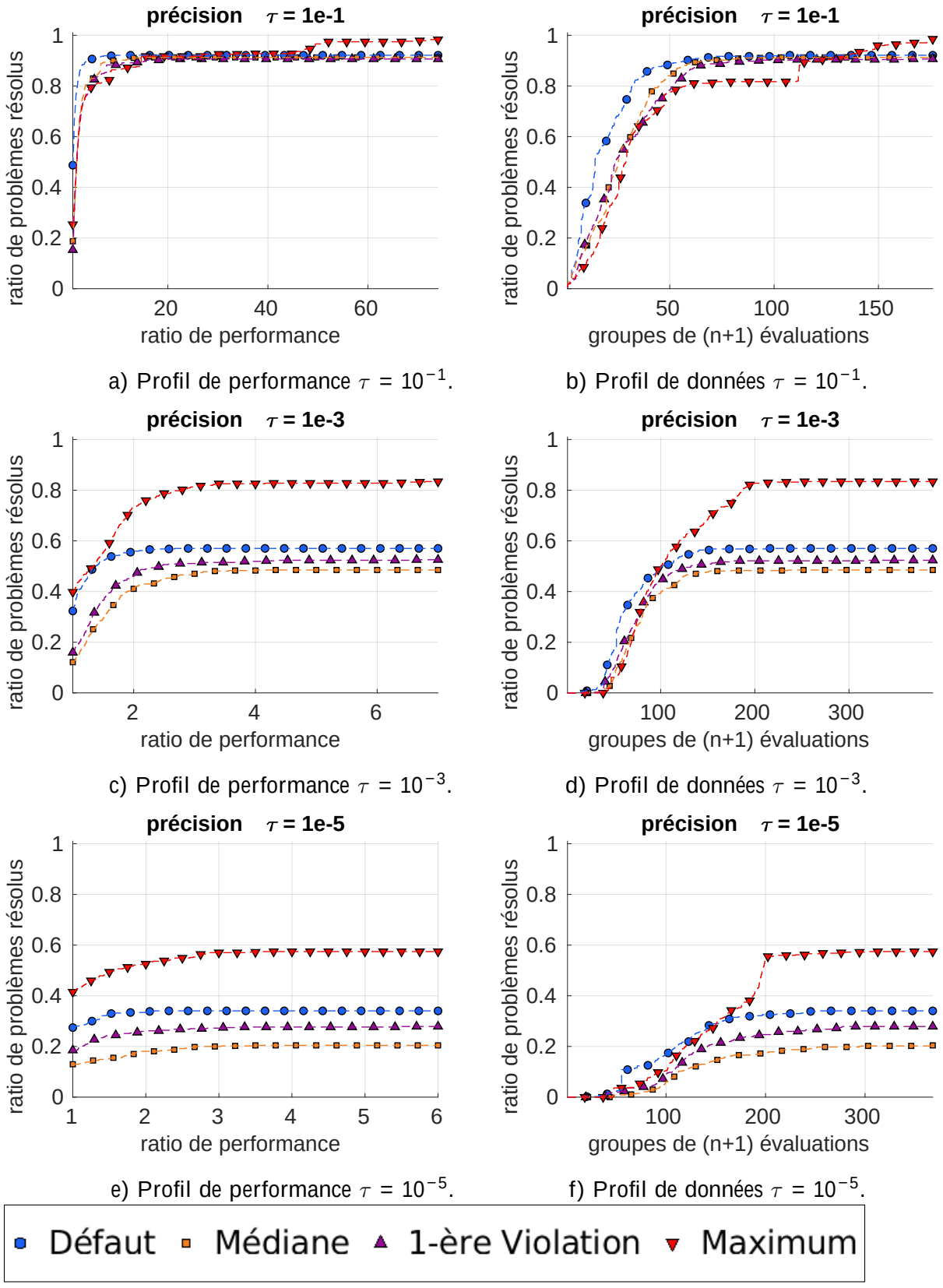
Tableau 5.1 Description de 5 problèmes analytiques de la littérature.

La figure 5.2 regroupe les profils de données et de performance regroupant les 500 instances sur des problèmes analytiques non modifiés. La pondération par le maximum domine assez largement les autres méthodes sur les profils. À la précision $\tau = 10^{-5}$, elle résout environ 58% des problèmes, quand le deuxième n'en résout qu'environ 35%.

La figure 5.3 regroupe les profils de données et de performance regroupant les 500 instances sur des problèmes analytiques «déséquilibrés». Les figures montrent une domination de la pondération par le maximum, à la fois pour les profils de données et de performance. Les deux autres pondérations semblent légèrement avoir de moins bons résultats que sans pondération.

Les problèmes analytiques montrent pour l'instant une nette domination de la pondération par le maximum.

Boites noires Les boites noires utilisées pour les tests numériques sont «STYRENE», «MDO» et «Lockwood» dont des descriptions se trouvent dans ces références [14, 28, 110]



a) Profil de performance $\tau = 10^{-1}$.

b) Profil de données $\tau = 10^{-1}$.

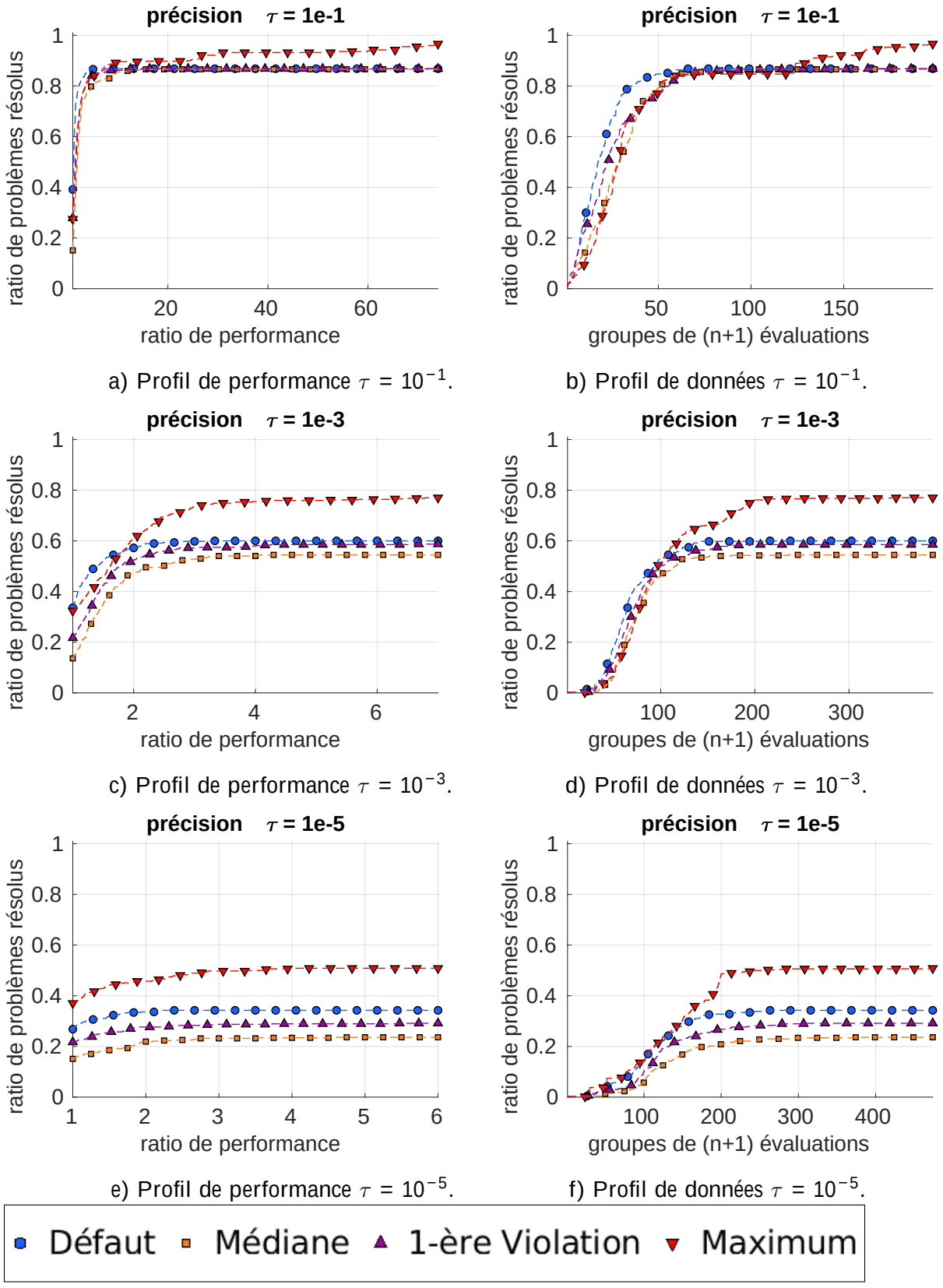
c) Profil de performance $\tau = 10^{-3}$.

d) Profil de données $\tau = 10^{-3}$.

e) Profil de performance $\tau = 10^{-5}$.

f) Profil de données $\tau = 10^{-5}$.

Figure 5.2 Profils de performance et de données $\tau = 10^{-1}$, $\tau = 10^{-3}$ et $\tau = 10^{-5}$ pour des problèmes analytiques non modifiés.



a) Profil de performance $\tau = 10^{-1}$.

b) Profil de données $\tau = 10^{-1}$.

c) Profil de performance $\tau = 10^{-3}$.

d) Profil de données $\tau = 10^{-3}$.

e) Profil de performance $\tau = 10^{-5}$.

f) Profil de données $\tau = 10^{-5}$.

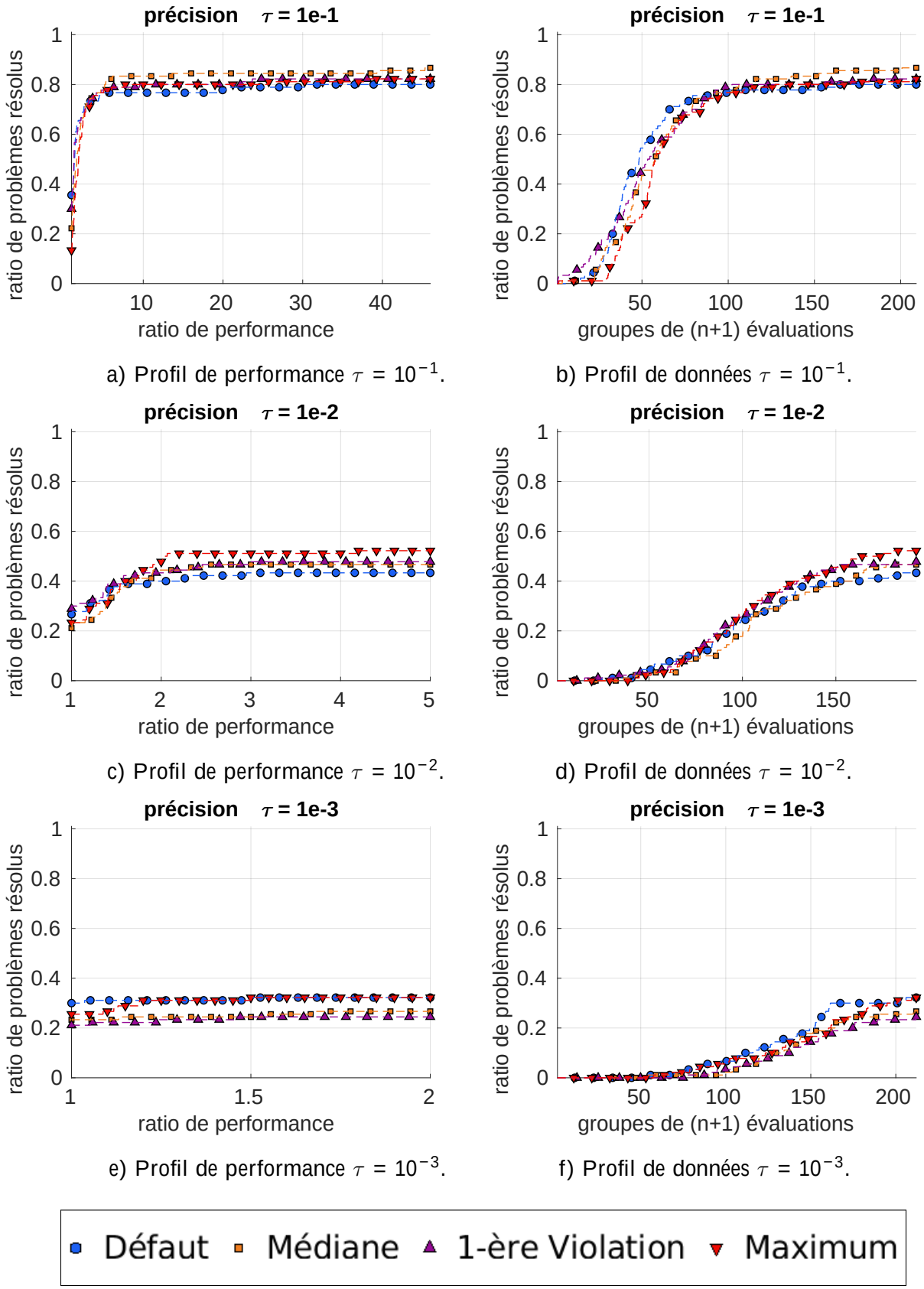
Figure 5.3 Profils de performance et de données $\tau = 10^{-1}$, $\tau = 10^{-3}$ et $\tau = 10^{-5}$ pour des problèmes analytiques «déséquilibrés».

ainsi qu'à le section 4.1. Pour chacune des boites noires, il y aura 30 points de départ réalisables qui seront utilisés.

La figure 5.4 présente les profils de données et de performance pour les 90 instances des trois boites noires non modifiés pour les précisions $\tau = 10^{-1}$, $\tau = 10^{-2}$ et $\tau = 10^{-3}$. À la précision $\tau = 10^{-1}$, la pondération par la médiane domine légèrement les autres. Aux deux autres précisions, la pondération qui domine est la pondération par le maximum. Cependant, à la précision $\tau = 10^{-3}$ ne domine pas la méthode par défaut.

On s'intéresse à présent aux instances des boites noires «déséquilibrées». Pour chacune des boites noires «déséquilibrées», il y a les mêmes points de départ que dans les versions non modifiées.

Ce que l'on observe sur les profils de la figure 5.5, c'est que la pondération par la première violation se détache comme étant celle qui est dominé par les autres. Ceci est notamment le cas aux précisions $\tau = 10^{-2}$ et $\tau = 10^{-3}$. Par rapport aux boites noires non modifiées, où elle avait eu de moins bons résultats que la méthode par défaut, la pondération par la médiane a des résultats équivalents à la méthode par défaut. Concernant la pondération par le maximum, elle a des résultats équivalents que la méthode par défaut pour la précision et $\tau = 10^{-3}$, mais se fait légèrement dominer les autres par la méthode par défaut pour $\tau = 10^{-2}$. Cette pondération domine cependant les autres pondérations aux précisions $\tau = 10^{-2}$ et $\tau = 10^{-3}$.



a) Profil de performance $\tau = 10^{-1}$.

b) Profil de données $\tau = 10^{-1}$.

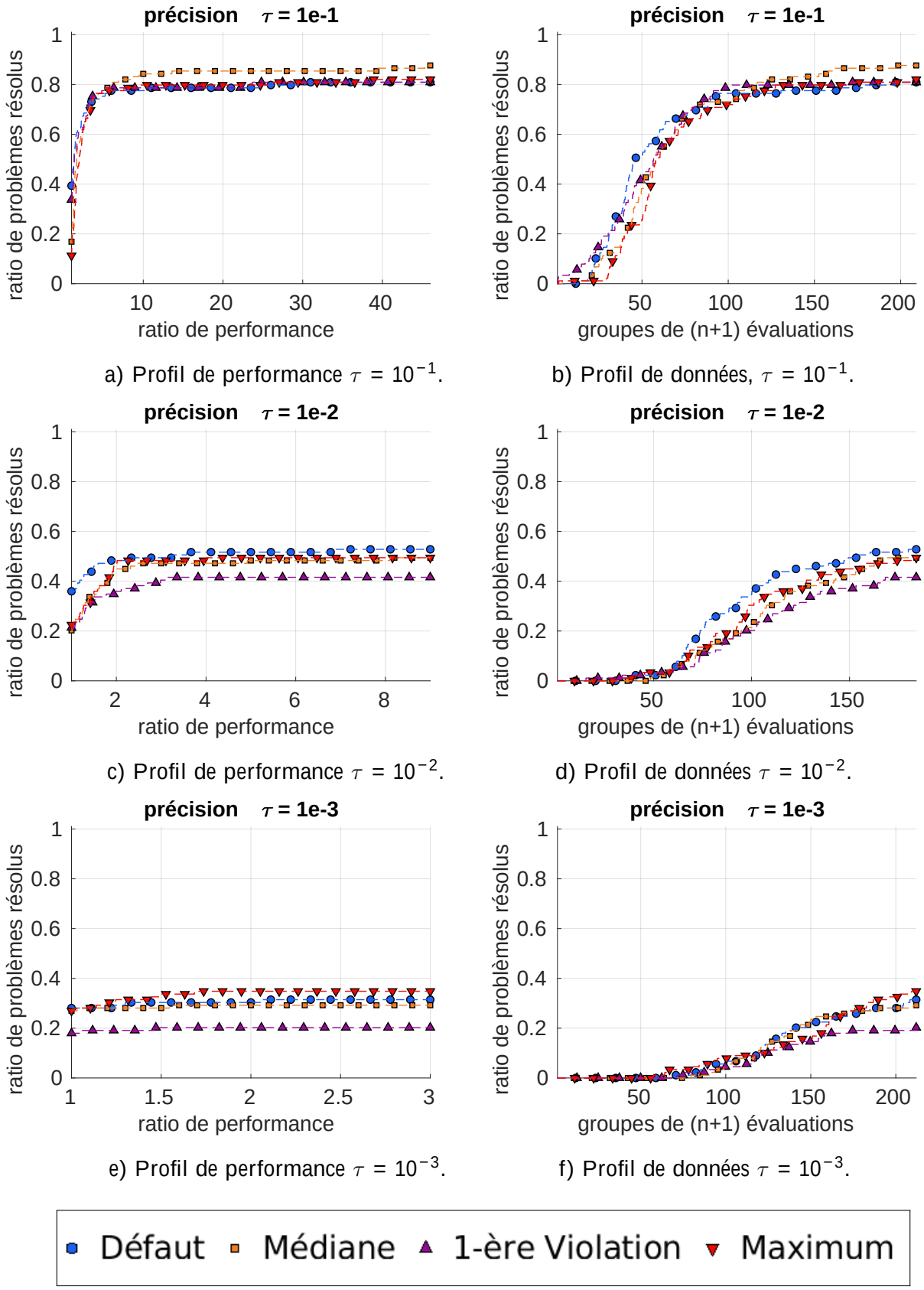
c) Profil de performance $\tau = 10^{-2}$.

d) Profil de données $\tau = 10^{-2}$.

e) Profil de performance $\tau = 10^{-3}$.

f) Profil de données $\tau = 10^{-3}$.

Figure 5.4 Profils de performance et de données $\tau = 10^{-1}$, $\tau = 10^{-2}$ et $\tau = 10^{-3}$ pour des boites noires non modifiées.



a) Profil de performance $\tau = 10^{-1}$.

b) Profil de données, $\tau = 10^{-1}$.

c) Profil de performance $\tau = 10^{-2}$.

d) Profil de données $\tau = 10^{-2}$.

e) Profil de performance $\tau = 10^{-3}$.

f) Profil de données $\tau = 10^{-3}$.

Figure 5.5 Profils de performance et de données avec $\tau = 10^{-1}$, $\tau = 10^{-2}$ et $\tau = 10^{-3}$ pour des boites noires «déséquilibrées».

5.4 Pondérations des contraintes relaxables pour $\tilde{h}$

Dans le cadre de la stratégie opportuniste, il est possible de faire un modèle ou une fonction substitut pour chaque contrainte, en utilisant des outils développés dans la sous-section 2.3.1. Pour tout $i \in \llbracket 1; m \rrbracket$, la fonction substitut de la contrainte c_i sera notée $\tilde{c_i}$. Dès lors, il est possible de faire une fonction substitut pour $\tilde{h}$ par

$$\tilde{h}(x) = \begin{cases} \sum_{i=1}^{m} \max\left(0, \tilde{c_i}(x)\right)^2 & \text{si } x \in X \\ +\infty & \text{sinon.} \end{cases}$$

Remarque : Dans cette section, on notera $\tilde{h}$, au lieu de $\tilde{h^k}$, pour rester cohérent avec les notations traditionnelles pour les fonctions substituts comme dans [8].

Actuellement, en supposant que l'on ait à disposition une fonction substitut $\tilde{f}$ pour f et $\tilde{h}$ pour h, l'ordonnancement suit la stratégie suivante : soient x et y, alors x est soumis à la boite noire avant y si et seulement si $(\tilde{f}(x) \leq \tilde{f}(y)$ et $\tilde{h}(x) < \tilde{h}(y))$ ou $(\tilde{f}(x) < \tilde{f}(y)$ et $\tilde{h}(x) \leq \tilde{h}(y))$.

Du fait que $\tilde{h}$ a de l'importance dans le fonctionnement de MADS et au regard des résultats de la pondération de h, une pondération de $\tilde{h}$ semble appropriée. On veut utiliser la même pondération pour h et pour $\tilde{h}$. Comme la pondération sur h avait montré la pertinence d'une pondération par le maximum, c'est donc celle-ci qui sera utilisée dans cette section. On choisira dès lors

$$\tilde{h}(x) = \begin{cases} \sum_{i=1}^{m} \max\left(0, \frac{\tilde{c_i}(x)}{a_i^k}\right)^2 & \text{si } x \in X \\ +\infty & \text{sinon.} \end{cases}$$

où la suite $(a_i^k)_{k \in \mathbb{N}}$ est définie par son terme général

$$a_i^k = \begin{cases} 1 & \text{si } \{c_i(x) : x \in \mathcal{C}^k, c_i(x) > 0\} = \emptyset \\ \max_{x \in \mathcal{C}^k} c_i(x) & \text{sinon.} \end{cases}$$

Ces pondérations peuvent être indépendantes. Il est possible de tester une pondération sur h, via $(h^k)_{k \in \mathbb{N}}$, sans en faire sur $\tilde{h}$ et vice versa. Enfin, il est possible de faire les deux pondérations en même temps. La pondération avec la $(h^k)_{k \in \mathbb{N}}$ seule ayant déjà été testée, les deux autres versions sont comparées avec NOMAD par défaut, mais avec les modèles activés.

5.4.1 Résultats numériques

Pour ces résultats numériques, trois versions vont être comparées. Elles utilisent toutes des modèles quadratiques pour la fonction objectif et les contraintes sur NOMAD 3.8.0 et en utilisant les directions générées par ORTHOMADS [60]. La première, représentée par les ronds dans les profils, est la méthode par défaut. La deuxième, représentée par des carrés, n'effectue une pondération que sur $\tilde{h}$. La troisième, représentée par des triangles, effectue des pondérations à la fois avec la suite $(h^k)_{k\ \mathbb{N}}$, mais aussi sur $\tilde{h}$. Les pondérations seront effectuées à l'aide de la violation maximum de chaque contrainte.

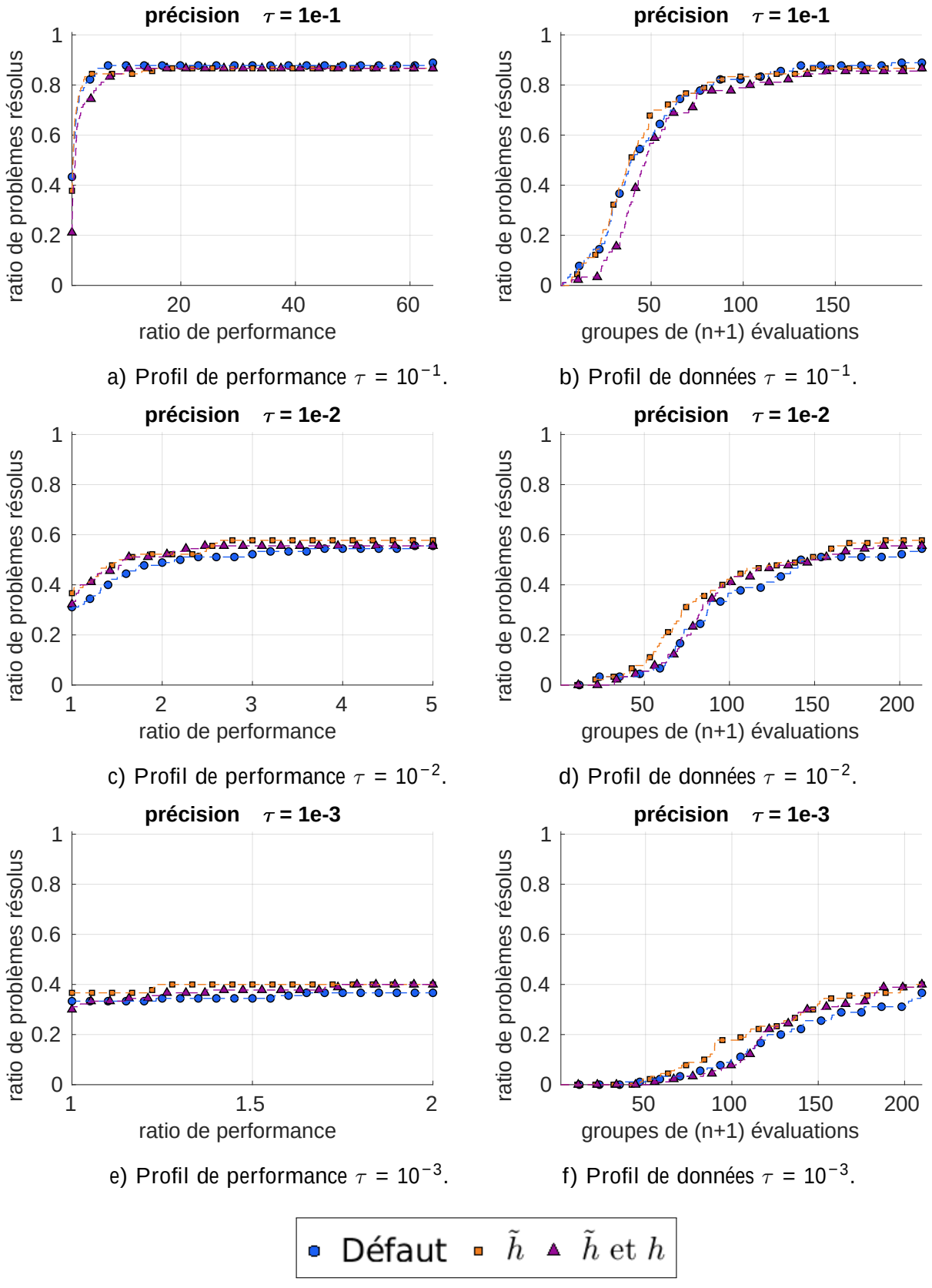
Les tests sont effectués sur les mêmes boites noires STYRENE, MDO et Lockwood qu'à la section 5.3.2.

La figure 5.6 s'intéresse aux boites noires non modifiées. À la précision $\tau = 10^{-1}$, les pondérations n'apportent aucune amélioration sur les boites noires testées. Aux précisions $\tau = 10^{-2}$ et $\tau = 10^{-3}$, paradoxalement, la pondération faite seulement sur $\tilde{h}$ apporte de meilleurs résultats qu'une pondération sur h et $\tilde{h}$ et la méthode par défaut. Cependant, les différences sont peu importantes et les résultats de la sous-section 5.3.2 avaient montré peu de résultats satisfaisants sur les boites noires non modifiées.

Les boites noires «déséquilibrées» sont également testées. Elles ont été déséquilibrées de la même façon qu'à la section 5.3.2.

La figure 5.7 s'intéresse aux boites noires «déséquilibrées». Ce qui est observé, c'est que la version où la pondération est faite sur h et sur $\tilde{h}$ domine les deux autres versions dans tous les cas. Il est intéressant de noter qu'aux précisions $\tau = 10^{-2}$, la pondération uniquement sur $\tilde{h}$ domine la version par défaut, mais se fait dominer par la troisième. Ceci montre l'effet cumulatif de la pondération des contraintes. S'il est préférable pour les problèmes mal échelonnés d'avoir une pondération sur h ou sur $\tilde{h}$, il est nettement préférable d'effectuer cette double pondération.

Par rapport à la figure 5.6, la figure 5.7 montre d'ailleurs l'efficacité de la pondération des contraintes sur des problèmes où les contraintes ne sont pas bien mises à l'échelle.



a) Profil de performance $\tau = 10^{-1}$.

b) Profil de données $\tau = 10^{-1}$.

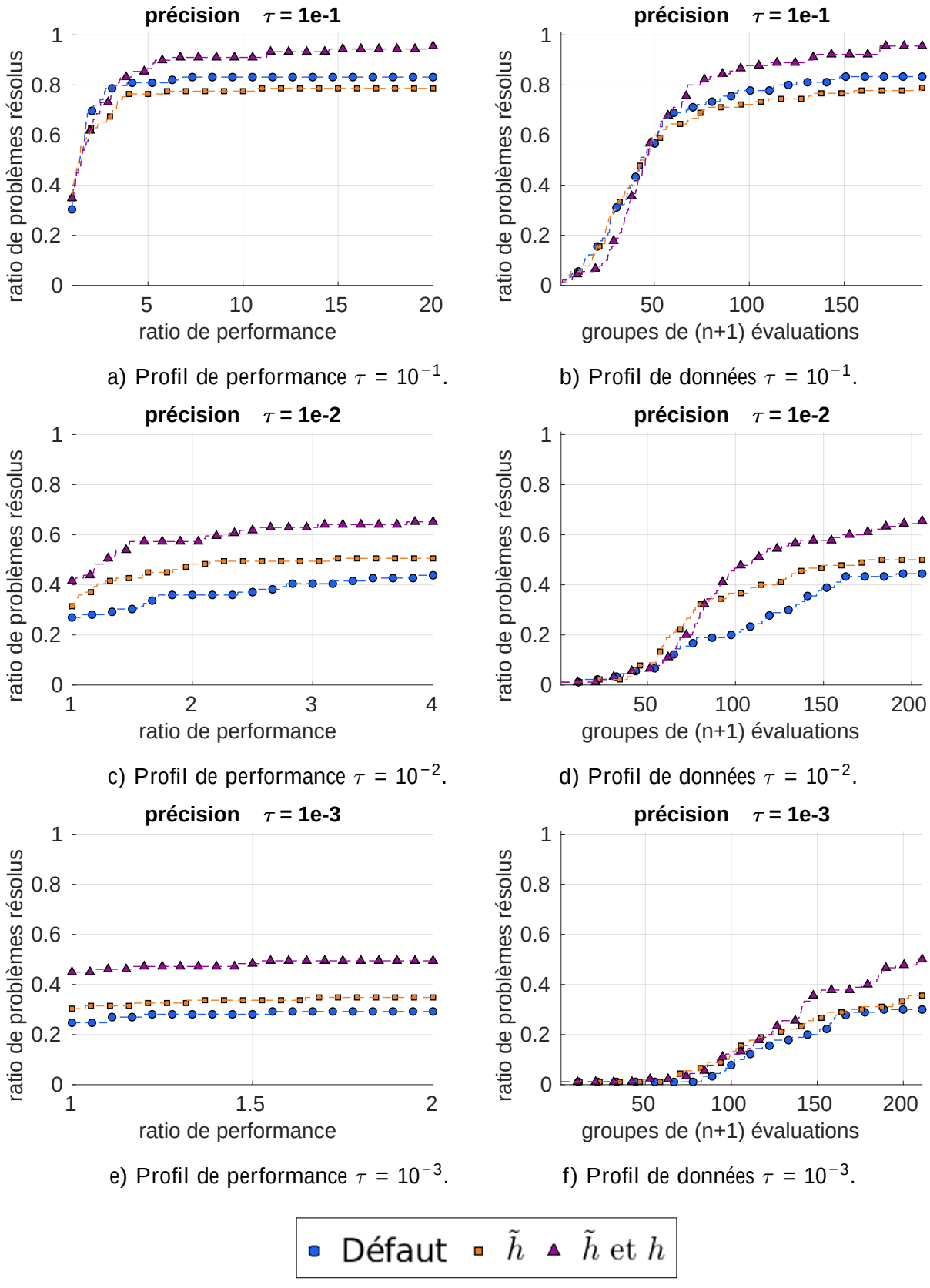
c) Profil de performance $\tau = 10^{-2}$.

d) Profil de données $\tau = 10^{-2}$.

e) Profil de performance $\tau = 10^{-3}$.

f) Profil de données $\tau = 10^{-3}$.

Figure 5.6 Profils de performance et de données avec $\tau = 10^{-1}$, $\tau = 10^{-2}$ et $\tau = 10^{-3}$ pour des boites noires non modifiées.



a) Profil de performance $\tau = 10^{-1}$.

b) Profil de données $\tau = 10^{-1}$.

c) Profil de performance $\tau = 10^{-2}$.

d) Profil de données $\tau = 10^{-2}$.

e) Profil de performance $\tau = 10^{-3}$.

f) Profil de données $\tau = 10^{-3}$.

Figure 5.7 Profils de performance et de données avec $\tau = 10^{-1}$, $\tau = 10^{-2}$ et $\tau = 10^{-3}$ pour des boites noires «déséquilibrées».

5.5 Discussion

Ce chapitre avait pour objectif de proposer plusieurs mises à l'échelle des contraintes et d'en étudier les impacts sur MADS en pondérant les contraintes. Ces pondérations reposaient sur les valeurs prises par les contraintes : la première violation, la médiane sur les n premières violations et la plus grande violation.

D'un point de vue théorique, l'analyse de convergence reposait sur la barrière progressive. Il a été montré pour les deux premières pondérations que l'analyse de convergence restait identique, à partir du rang où toutes les pondérations étaient faites. Pour la troisième pondération, les propriétés de $\{h^k_{max}\}_{k \in \mathbb{N}}$ sont conservées.

Concernant les résultats numériques, il a pu être mis en évidence qu'une pondération a un impact positif sur les problèmes qui ne sont pas correctement mis à l'échelle. La pondération n'a pas d'impact négatif quand le problème initial n'a pas de problème de mise à l'échelle. Parmi les pondérations proposées, celle qui semble avoir les meilleurs résultats est la pondération par la plus grande violation.

CHAPITRE 6 ESTIMATION DE LA FONCTION DE VIOLATION DE CONTRAINTES AVEC CONTRAINTES BINAIRES ET CONTRAINTES NON-RELAXABLES

Soient $X \subseteq \mathbb{R}^n$ et $l, u \in \mathbb{R}^n$. Considérons le problème d'optimisation avec contraintes suivant :

$$\begin{cases} \min\limits_{x \ X} f(x) \\ \text{sous contraintes } b_i(x) \leq 0, \ \forall i \in \llbracket 1, q \rrbracket \\ \qquad\qquad\qquad\quad c_i(x) \leq 0, \ \forall i \in \llbracket 1, m \rrbracket \\ \qquad\qquad\qquad\quad l \leq x \leq u. \end{cases}$$

où pour tout $i \in \llbracket 1, q \rrbracket$, $b_i : \mathbb{R}^n \mapsto \{0; 1\}$, pour tout $i \in \llbracket 1, m \rrbracket$, $c_i : \mathbb{R}^n \mapsto \mathbb{R}$ et $f : \mathbb{R}^n \mapsto \mathbb{R}$. Dans ce chapitre, et à l'inverse du chapitre 2, l'ensemble X contient toutes les contraintes non-relaxables mais aussi les contraintes cachées. Les contraintes cachées sont celles qui ne sont pas connues de l'utilisateur [9]. Les contraintes, pour tout $i \in \llbracket 1, q \rrbracket$, $b_i(x) \leq 0$ et, pour tout $i \in \llbracket 1, m \rrbracket$, $c_i(x) \leq 0$ sont des contraintes relaxables. Les contraintes non-relaxables et les contraintes cachées sont actuellement traitées par la barrière extrême. L'article [121] avait proposé un traitement des contraintes cachées en modifiant l'algorithme nommé «DIRECT». Les contraintes binaires sont traitées de par leur nature par la barrière extrême. Ce chapitre propose à la fois un traitement pour un problème avec plusieurs contraintes binaires et un nouveau traitement des contraintes non-relaxables et cachées. Les contraintes binaires seront modélisées à l'aide de k-nn (voir la section 3.2.1). Ces modèles serviront pour estimer la fonction de violations des contraintes. Les contraintes non-relaxables et cachées seront traitées comme des contraintes binaires.

6.1 Traitement des contraintes binaires et des contraintes non-relaxables

Les contraintes non-relaxables et cachées seront jumelées afin de ne former qu'une seule contrainte. Dès lors, le fait de fusionner toutes ces contraintes la rendra forcément non-quantifiable, comme cela est décrit dans [9]. La contrainte est soit satisfaite, soit elle ne l'est pas, mais sans qu'il soit possible de donner une information plus précise. Ainsi, fusionner ces contraintes en une seule contrainte binaire a un sens. Pour cela, la contrainte $x \in X$ sera remplacée par la contrainte $b_0(x) \leq 0$ où $b_0 : \mathbb{R}^n \mapsto \{0; 1\}$ avec $b_0 = 1 - \mathbb{1}_X$. La fonction $\mathbb{1}_{\mathbb{X}}$

est définie par :

$$\mathbb{1}_X(x) = \begin{cases} 1 & \text{si } x \in X \\ 0 & \text{si } x \notin X. \end{cases}$$

Le problème possède la reformulation suivante :

$$\begin{cases} \min\limits_{x \ \mathbb{R}^n} f(x) \\ \text{sous contraintes } b_i(x) \leq 0, \ \forall i \in \llbracket 0, q \rrbracket \\ \qquad\qquad\qquad\qquad c_i(x) \leq 0, \ \forall i \in \llbracket 1, m \rrbracket \\ \qquad\qquad\qquad\qquad l \leq x \leq u. \end{cases}$$

Dans la section 2.3.2, il a été écrit que la barrière progressive avait permis une gestion plus souple des contraintes que la barrière extrême. La barrière progressive utilise la fonction de violation des contraintes, mais uniquement de celles qui sont relaxables. Donc la formulation de celle-ci reste inchangée malgré l'apparition de la nouvelle contrainte $b_0(x) \leq 0$. Pour tout $x \in \mathbb{R}^n$,

$$h(x) = \begin{cases} \sum\limits_{i=1}^{q} \max(0, b_i(x))^2 + \sum\limits_{i=1}^{m} \max(0, c_i(x))^2 & \text{si } b_0(x) \leq 0 \\ +\infty & \text{sinon.} \end{cases}$$

qui peut se ré-écrire encore plus simplement en profitant du fait que pour tout $i \in \llbracket 1; q \rrbracket$, $b_i : \mathbb{R}^n \mapsto \{0; 1\}$ par, pour tout $x \in \mathbb{R}^n$

$$h(x) = \begin{cases} \sum\limits_{i=1}^{q} b_i(x) + \sum\limits_{i=1}^{m} \max(0, c_i(x))^2 & \text{si } b_0(x) \leq 0 \\ +\infty & \text{sinon.} \end{cases}$$

Il peut être souligné que dans ce cadre, des problèmes de mise à l'échelle peuvent intervenir étant donné que les fonctions c_i ne sont pas forcément à la même échelle que les contraintes binaires. Ceci sera étudié en section 6.2
De nouvelles façons de créer une fonction substitut (voir 2.3.1) de h vont être proposées. Celles-ci sont définies, pour tout $x \in \mathbb{R}^n$, par

$$\tilde{h_1}(x) = \sum_{i=0}^{q} \max(0, \tilde{b_i}(x))^2 + \sum_{i=1}^{m} \max(0, \tilde{c_i}(x))^2. \tag{6.1}$$

$$\tilde{h_2}(x) = \sum_{i=0}^{q} \max(\tilde{b_0}(x), \tilde{b_i}(x))^2 + \sum_{i=1}^{m} \max(0, \tilde{c_i}(x))^2. \tag{6.2}$$

$$\tilde{h_3}(x) = (1 + \tilde{b_0}(x)) \left(\sum_{i=1}^{q} \max(0, \tilde{b_i}(x))^2 + \sum_{i=1}^{m} \max(0, \tilde{c_i}(x))^2 \right). \tag{6.3}$$

L'équation 6.1 est la prolongation naturelle de la version par défaut de $\tilde{h}$ en prenant en compte les contraintes binaires, avec des modèles de contraintes binaires $\tilde{b_i}$, $i \in \llbracket 1; q \rrbracket$, et en prenant en compte les contraintes non-relaxables et binaires, via $\tilde{b_0}$.

L'équation 6.2 est très proche de 6.1 en remplaçant le terme $\max(0, \tilde{b_i}(x))^2$ par $\max(\tilde{b_0}(x), \tilde{b_i}(x))^2$. Ceci a pour but de favoriser l'obtention des points réalisant la contrainte b_0.

L'équation 6.3 est la prolongation naturelle de la version par défaut de $\tilde{h}$ et ce en faisant une pénalisation multiplicative pour la contrainte b_0. À noter que si l'on choisit k-nn avec le noyau d'Epanechnikov (comme au chapitre 4), alors pour $x \in \mathbb{R}^n$, $(1 + \tilde{b_0}(x)) \in [1; 2]$. Si l'on pense que les contraintes non-relaxables et cachées ont de fortes chances d'être satisfaites par x, alors $\tilde{b_0}(x)$ est proche de 0, donc $(1 + \tilde{b_0}(x))$ est proche de 1. Ainsi le facteur de droite n'est pas pénalisé. À l'inverse, si l'on pense que les contraintes non-relaxables et cachées ont peu de chances d'être satisfaites par x, alors $\tilde{b_0}(x)$ est proche de 1, donc $(1 + \tilde{b_0}(x))$ est proche de 2. Ainsi le facteur de droite est doublé, ce qui le pénalise davantage.

Avec $\tilde{f}$ et $\tilde{h}$, NOMAD trie les points en se conformant à la règle suivante : soient x et y, alors x est soumis à la boite noire avant y si et seulement si $(\tilde{f}(x) \leq \tilde{f}(y)$ et $\tilde{h}(x) < \tilde{h}(y))$ ou $(\tilde{f}(x) < \tilde{f}(y)$ et $\tilde{h}(x) \leq \tilde{h}(y))$.

6.1.1 Résultats numériques

STYRENE est un problème comprenant naturellement quatre contraintes binaires et des contraintes cachées. Les contraintes cachées ont un poids important, puisqu'il y a approximativement 14% des points testés qui violent une contrainte cachée [58]. Ce problème n'a subi aucune modification avant d'effectuer les tests. MDO ne possède initialement ni de contrainte binaire, ni de contrainte cachée ou non-relaxable. Pour observer un traitement de toutes ces

contraintes, la première contrainte est traitée comme une contrainte binaire et la deuxième comme une contrainte non-relaxable ; la dernière restant traitée par la barrière progressive. Comme pour MDO, Lockwood voit sa première contrainte traitée comme une contrainte binaire et sa deuxième comme une contrainte non-relaxable.

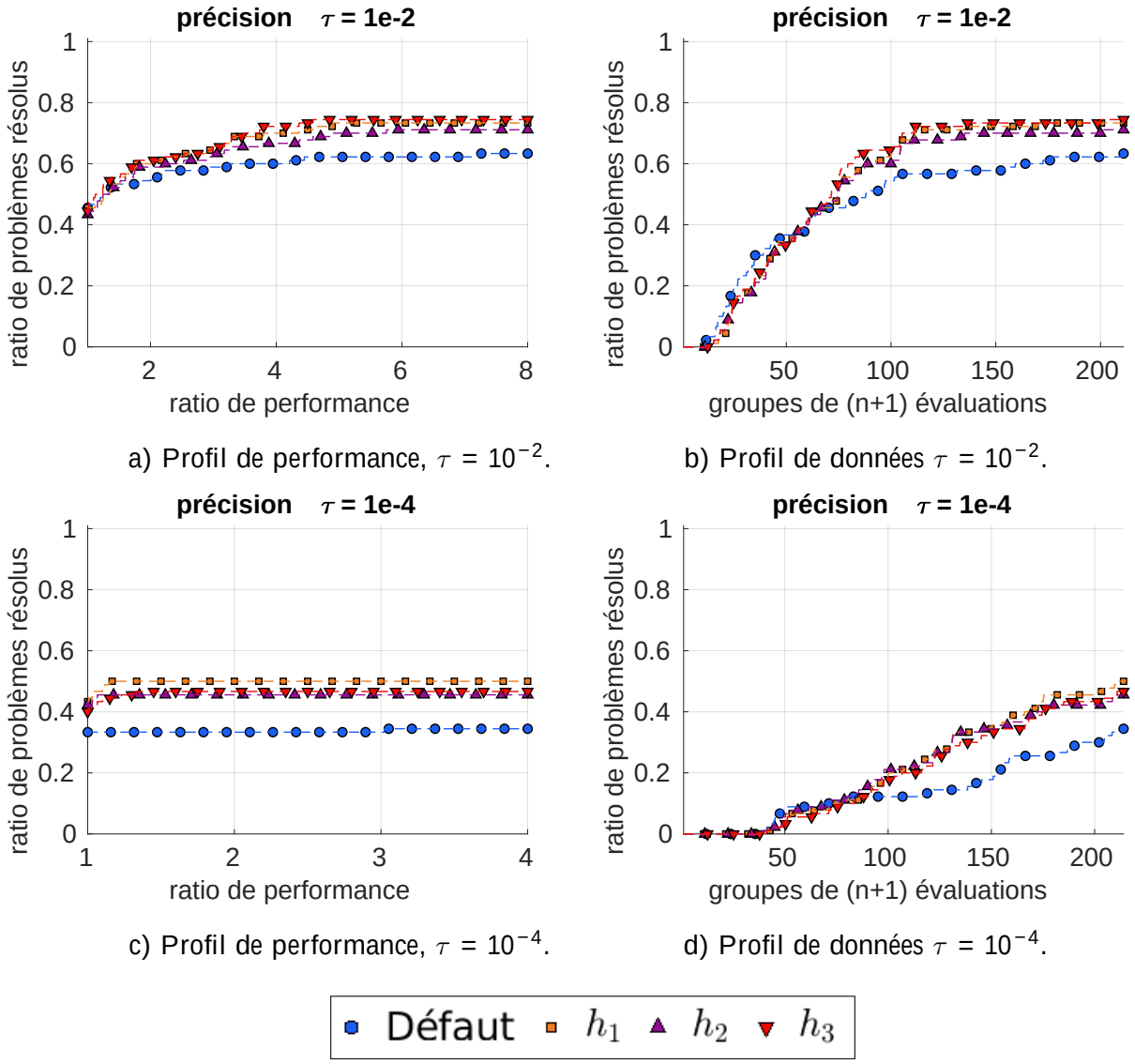
Les tests ont été effectués sur NOMAD 3.8.0 utilisant ORTHOMADS [60], des modèles quadratiques et un budget de 1500 évaluations. Les fonctions substituts $\tilde{b}_i$, $i \in \llbracket 1, q \rrbracket$, sont déterminées en utilisant k-nn, et avec le noyau d'Épanechnikov avec comme nombres de voisins $\sqrt{|\mathcal{C}^k|}$, où $k \in \mathbb{N}$ est le numéro de l'itération. Des tests ont été faits avec LOOCV et $10 - fold - CV$ 3.1, mais sur ces boites noires, du fait de la pluralité de validations croisées requises, les temps de calcul étaient trop longs. Ceci était le cas malgré le fait que les algorithmes d'optimisation de boites noires utilisent plus souvent le nombre d'évaluations que le temps de calcul comme ressources disponibles. C'est pour cette raison qu'une règle plus simple dans le choix du nombre des voisins ($\sqrt{|\mathcal{C}^k|}$) a été utilisée.

La figure 6.1 montre que chacun des trois nouveaux traitements de $\tilde{h}$ résout plus de problèmes que la méthode par défaut. Bien que l'on observe peu de différences entre chacun des traitements, il semble cependant que le traitement avec la fonction h_1 entraine de meilleurs résultats.

Il est possible de se demander s'il existe un problème de différences d'échelle des contraintes c_i par rapport aux contraintes binaires. Les contraintes binaires sont toutes à l'échelle «1» les unes par rapport aux autres du fait qu'elles ne prennent toutes que 0 et 1 comme valeurs. La section 6.2 traite de cette problématique à l'aide des résultats du chapitre 5.

6.2 Mise à l'échelle des valeurs des contraintes

Le chapitre 5 a mis en évidence l'intérêt de mettre à l'échelle les contraintes. L'idée est d'utiliser les conclusions du chapitre 5 pour donner des poids aux estimations de chaque contrainte dans le cas où il y a des contraintes binaires. La mise à l'échelle sera dépendante de l'itération $k \in \mathbb{N}$. Le coefficient utilisé pour la mise à l'échelle de la contrainte c_i sera noté a_i^k. Sa définition prend en compte les conclusions du chapitre 5 et vaut,



a) Profil de performance, $\tau = 10^{-2}$.

b) Profil de données $\tau = 10^{-2}$.

c) Profil de performance, $\tau = 10^{-4}$.

d) Profil de données $\tau = 10^{-4}$.

Figure 6.1 Profils de performance et de données avec $\tau = 10^{-2}$ et $\tau = 10^{-4}$ pour des boites noires.

$$a_i^k = \begin{cases} 1 & \text{si } \{c_i(x) : x \in \mathcal{C}^k, c_i(x) > 0\} = \emptyset \\ \max\limits_{x \ \mathcal{C}^k} c_i(x) & \text{sinon.} \end{cases}$$

où $\mathcal{C}^k$ est la cache au début de l'itération k.

Il n'y a pas besoin de calculer les pondérations pour les contraintes binaires, car, avec la même méthode de calcul, ils vaudraient tous 1.

Les versions qui seront proposées auront des formulations similaires à la section 6.1. Voici des nouvelles estimations de la fonction de violations de contraintes qui prennent en compte les contraintes binaires, non-relaxables et cachées. Elles sont définies pour tout $x \in \mathbb{R}^n$ par

$$\tilde{h_4}(x) = \sum_{i=0}^{q} \max\left(0, \tilde{b_i}(x)\right)^2 + \sum_{i=1}^{m} \max\left(0, \frac{\tilde{c_i}(x)}{a_i^k}\right)^2. \tag{6.4}$$

$$\tilde{h_5}(x) = \sum_{i=0}^{q} \max\left(\tilde{b_0}(x), \tilde{b_i}(x)\right)^2 + \sum_{i=1}^{m} \max\left(0, \frac{\tilde{c_i}(x)}{a_i^k}\right)^2. \tag{6.5}$$

$$\tilde{h_6}(x) = (1 + \tilde{b_0}(x)) \left(\sum_{i=1}^{q} \max\left(0, \tilde{b_i}(x)\right)^2 + \sum_{i=1}^{m} \max\left(0, \frac{\tilde{c_i}(x)}{a_i^k}\right)^2\right). \tag{6.6}$$

Enfin, du fait que les contraintes soient mises à l'échelle, on peut en profiter pour adapter $\tilde{h_5}$.

$$\tilde{h_7}(x) = \sum_{i=0}^{q} \max\left(\tilde{b_0}(x), \tilde{b_i}(x)\right)^2 + \sum_{i=1}^{m} \max\left(\tilde{b_0}(x), \frac{\tilde{c_i}(x)}{a_i^k}\right)^2. \tag{6.7}$$

De fait, les contraintes sont davantage comparables entre elles, notamment avec les contraintes binaires.

Résultats numériques Les résultats numériques présentés ici sont obtenus en comparant la méthode par défaut NOMAD 3.8.0 et celles qui proposent un traitement des contraintes binaires, non-relaxables et cachées en plus d'une mise à l'échelle des contraintes. Les directions seront générées par ORTHOMADS [60] et les modèles quadratiques sont activés. Les tests sont effectués sur 3 boites noires : STYRENE, MDO et Lockwood [14, 28, 110]. Pour chacun des problèmes, il y aura des tests sur 30 instances partant de points de départ réalisables, soit

90 instances au total. Il s'agit des 90 mêmes instances et des mêmes paramètres de NOMAD qu'à la section 6.1.

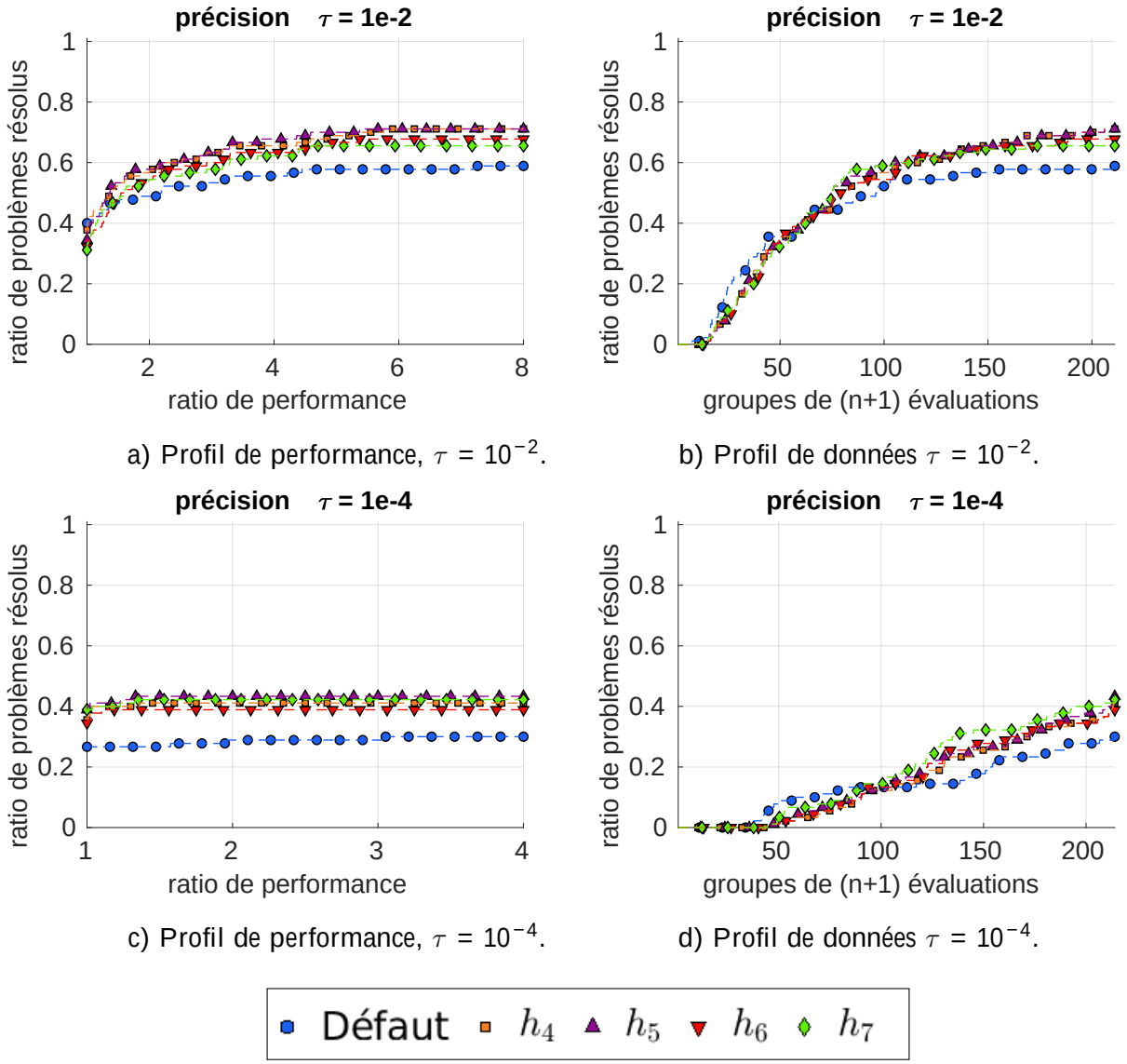
Les $\tilde{b}_i$, $i \in \llbracket 1, q \rrbracket$, sont déterminés en utilisant k-nn avec le noyau d'Épanechnikov avec comme nombres de voisins $\sqrt{|\mathcal{C}^k|}$, où $k \in \mathbb{N}$ est le numéro de l'itération.

On va présenter les résultats sur des profils de données et de performance aux précisions $\tau \in \{10^{-2}, 10^{-4}\}$ sur les 90 instances des 3 boites noires en même temps.

La figure 6.2 compare les versions utilisant la mise à l'échelle entre elles avec la version par défaut. On observe que les quatre versions dominent toutes la version par défaut sur les profils de performance. Sur les deux profils de données, on observe qu'au début de la convergence, la méthode par défaut domine. Cependant, les courbes se croisent à environ $50 \times (n+1)$ à $\tau = 10^{-2}$ et $100 \times (n+1)$ à $\tau = 10^{-4}$. Il est à souligner que la version h_7 se fait dominer par rapport aux traitements des contraintes à la précision 10^{-2}, mais semble fournir de meilleurs résultats que les autres à 10^{-4}. Celle qui semble le mieux fonctionner, et concurrence le mieux h_7 à la précision $\tau = 10^{-4}$, est en moyenne h_5. Rappelons que h_5 et h_7 avaient des définitions très proches. La version h_7 profitait davantage de la mise à l'échelle. Il peut donc être préférable de la favoriser malgré tout.

Il est à souligner qu'il y a peu de différences en terme de ratio de problèmes résolus entre les versions avec et sans mise à jour. Il est donc pertinent d'essayer de dégager l'une des sept versions par rapport aux autres. Pour cela, la meilleure version sans mise à l'échelle (h_1) et la meilleure version avec mise à l'échelle (h_7) vont être comparées.

La figure 6.3 montre des profils de données et de performance pour comparer h_1 et h_7 sur les 90 instances étudiées dans ce chapitre. Le budget est toujours de 1500 évaluations. Les modèles quadratiques sont activés. La version qui doit être préférée est h_7. Celle-ci domine légèrement la version h_1 sur les profils de performance à la précision $\tau = 10^{-2}$ et $\tau = 10^{-4}$ et le profil de données à la précision $\tau = 10^{-2}$. Le profil de données à la précision $\tau = 10^{-4}$ montre des résultats similaires entre h_1 et h_7.



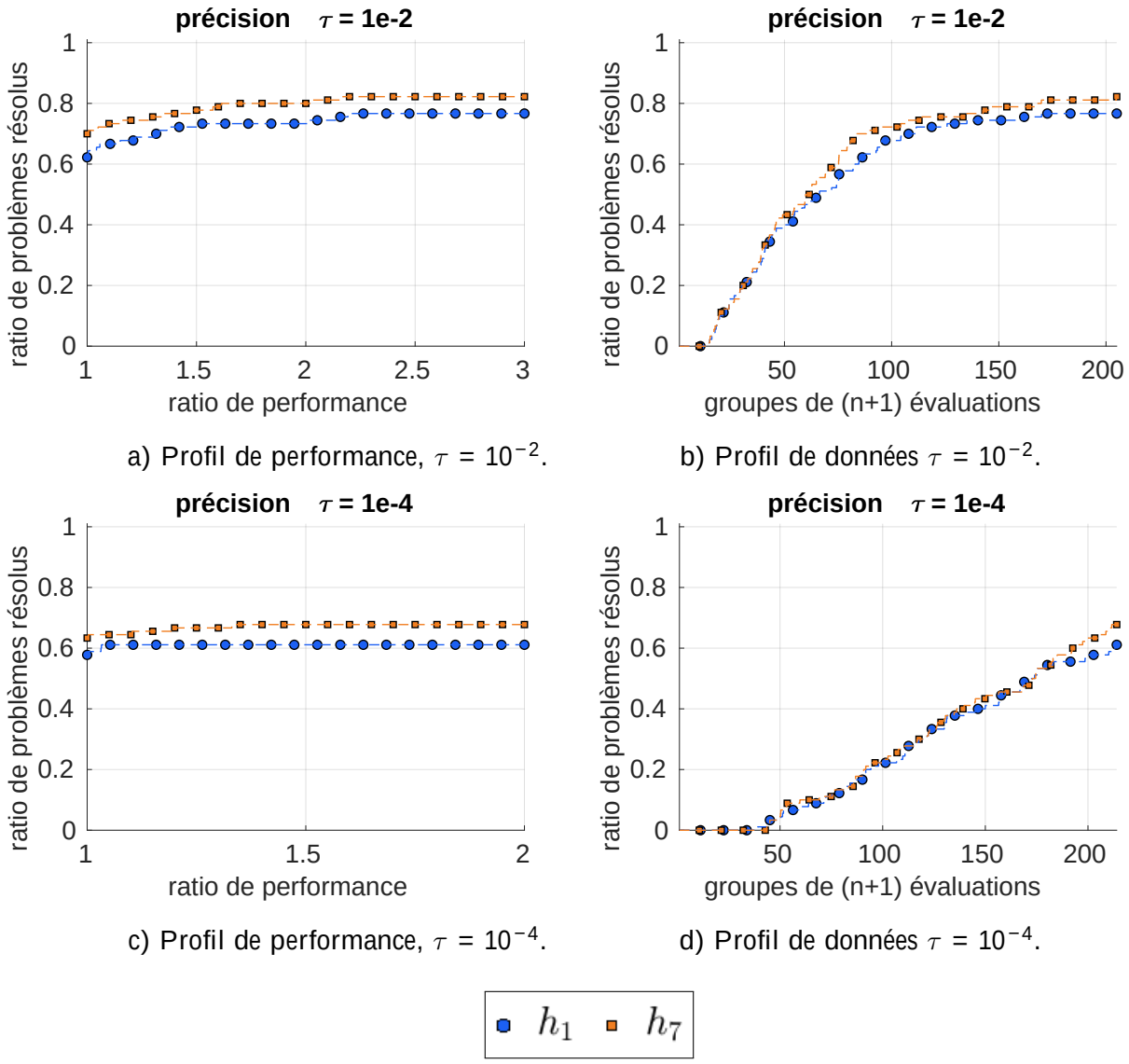
a) Profil de performance, $\tau = 10^{-2}$.

b) Profil de données $\tau = 10^{-2}$.

c) Profil de performance, $\tau = 10^{-4}$.

d) Profil de données $\tau = 10^{-4}$.

Figure 6.2 Profils de performance et de données avec $\tau = 10^{-2}$ et $\tau = 10^{-4}$ pour des boites noires.



a) Profil de performance, $\tau = 10^{-2}$.

b) Profil de données $\tau = 10^{-2}$.

c) Profil de performance, $\tau = 10^{-4}$.

d) Profil de données $\tau = 10^{-4}$.

Figure 6.3 Profils de performance et de données avec $\tau = 10^{-2}$ et $\tau = 10^{-4}$ pour des boites noires pour les deux meilleures versions.

6.3 Discussion

Dans ce chapitre, l'objectif était de proposer un traitement pour plusieurs contraintes : binaires, non-relaxables et cachées. Les contraintes non-relaxables et cachées ont été transformées en une contraintes binaire. Le traitement des contraintes binaires a été fait par le calcul d'un modèle $\tilde{h}$ de la fonction de violation de contraintes. Plusieurs variantes ont été proposées, notamment concernant l'utilisation de la contrainte binaire regroupant les contraintes non-relaxables et cachées.

Des premiers résultats numériques ont été effectués sans mise à l'échelle des contraintes. Ceci avait pour but d'isoler l'importance d'un traitement de ces contraintes. Les différentes versions ont toutes montré qu'un traitement pour les contraintes binaires, non-relaxables et cachées a un aspect positif, car elles ont toutes dominé la version de MADS par défaut. Chaque traitement a dominé la méthode par défaut. Il existe cependant peu de différences entre chacun des trois traitements. S'il faut en choisir un, la version avec h_1 a montré une très légère supériorité.

Ensuite, du fait qu'une contrainte binaire a une échelle qui ne dépend pas de la formulation de l'utilisateur, elle risque de ne pas être à l'échelle par rapport aux autres contraintes. Ainsi, les conclusions du chapitre 5 ont été utilisées pour voir si une pondération des autres contraintes ne donnerait pas un traitement plus équitable des contraintes binaires. Ceci n'a été testé que sur des boites noires qui n'ont pas été déséquilibrées. Peu de différences ont été observées par rapport aux tests précédents. Cependant, la version h_7, favorisant le fait que les contraintes non-relaxables et cachées soient vérifiées, a montré les résultats les plus satisfaisants. Cette dernière est celle recommandée si l'on veut effectuer une mise à l'échelle des contraintes. Sans cette mise à l'échelle, soit on peut choisir la version h_2 (si on veut trouver la formulation la plus proche de h_7), soit on peut choisir la version h_1 (car c'est elle qui a fourni de meilleurs résultats sans mise à l'échelle).

Il a été considéré que la contrainte $x \in X$ était modélisée par l'entremise d'une seule contrainte. Cependant, lorsqu'il y a plusieurs contraintes non-relaxables identifiées, avoir une contrainte binaire par contrainte non-relaxable pourrait être envisagé. Cela nécessiterait des ajustements par rapport à ce chapitre.

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS

Dans cette thèse, plusieurs extensions de l'algorithme MADS ont été développées. Celles-ci ont été réalisées dans un cadre d'optimisation de boites noires. Ces extensions ont concerné l'ordonnancement des points, dans le cadre de la stratégie opportuniste, et la gestion des contraintes. Des stratégies d'ordonnancement ont été proposées dans un contexte sans modèle dans le chapitre 4. Le chapitre 5 a montré que lorsque les contraintes n'étaient pas correctement à l'échelle, alors une mise à l'échelle de celles-ci permettaient d'améliorer la performance de MADS. Le chapitre 6 a permis de définir un nouveau traitement des contraintes binaires, non-relaxables et cachées. Ce traitement, basé sur la création de modèles de ces contraintes, produit de meilleurs résultats qu'avec le traitement par défaut, utilisant la barrière extrême. Ceci renforce l'idée qu'il est toujours utile d'avoir des modèles des fonctions mises en jeu dans un problème d'optimisation de boites noires.

7.1 Synthèse des travaux et limites

Le chapitre 4 avait pour objectif de proposer de nouvelles stratégies d'ordonnancement pouvant améliorer celle qui est proposée par défaut dans MADS en l'absence de modèles. Deux stratégies très différentes ont été testées. L'une d'entre elles cherchait à évaluer prioritairement les points qui avaient le plus de chances d'être réalisables. Pour cela, on estimait la probabilité qu'un point soit réalisable à l'aide de l'algorithme k-nn. Plusieurs noyaux ont été testés ; le noyau d'Épanechnikov avec LOOCV est celui qui a obtenu les meilleurs résultats. Cette estimation avec k-nn n'est cependant pas parvenue à battre la stratégie d'ordonnancement basée sur la direction de dernier succès, la stratégie par défaut dans NOMAD sans modèles. Il a pu être mis en évidence, notamment avec le problème SNAKE, que la stratégie basée sur la potentielle réalisabilité est trop prudente. Une deuxième stratégie, plus aventureuse, proposait d'évaluer les points d'abord les plus éloignés par rapport à la cache. Ceci a montré des améliorations sur des problèmes de boites noires. Il peut donc être recommandé d'utiliser cette stratégie dans NOMAD par défaut, à la place de la direction de dernier succès, dans le cadre de problèmes sans modèles.

Le chapitre 5 s'intéressait aux questions de mise à l'échelle dans MADS. Il a été montré que des problèmes équivalents ne subissent pas les mêmes traitements dans MADS, car des différences d'échelles créent des fonctions de violations de contraintes différentes. Ces diffé-

rences peuvent notamment donner un poids arbitrairement élevé ou faible à une contrainte par rapport à d'autres. Ces différences ont un impact à la fois sur la barrière progressive, via la fonction de violations de contraintes h, et sur l'ordonnancement de points candidats, via l'estimation de la fonction de violations de contraintes $\tilde{h}$. Plusieurs contributions ont été faites à ce sujet. Concernant l'impact sur la barrière progressive, au lieu d'avoir une seule fonction de violations de contraintes, une suite $\{h^k\}_{k\ \mathbb{N}}$ de fonctions de violations de contraintes a été créée pour mettre à l'échelle les contraintes. Une autre contribution a concerné les changements occasionnés sur la barrière progressive, sur les points potentiellement dominés et sur le choix de la barrière h_{max}. De plus, plusieurs points de convergence selon les différentes pondérations ont été obtenus. Une pondération sur h a montré des résultats positifs, principalement sur des boites noires déséquilibrées, avec une pondération par la plus grande violation. Ceci a permis de montrer que, si un problème possède une mauvaise mise à l'échelle, alors les performances peuvent être améliorées en pondérant les contraintes. Différentes mises à l'échelle ont également été proposées pour $\tilde{h}$. Les résultats numériques ont montré un impact, en effectuant des tests sur des boites noires déséquilibrées, en pondérant $\tilde{h}$. Enfin, combiner des mises à l'échelle pour h et $\tilde{h}$ a permis d'avoir des résultats meilleurs qu'en ne pondérant que $\tilde{h}$. Il est donc recommandé d'effectuer cette opération pour tous les problèmes car, si le problème est déséquilibré alors l'impact est positif, et s'il est correctement mis à l'échelle, l'impact n'est pas négatif ; donc cette mise à l'échelle ne nuira pas en moyenne.

Le chapitre 6 avait pour objectif de proposer un nouveau traitement de plusieurs contraintes : binaires, non-relaxables et cachées. Ces types de contraintes utilisaient auparavant la barrière extrême. Il y a plusieurs contributions dans ce chapitre. La première a été de regrouper les contraintes non-relaxables et cachées en une nouvelle contrainte binaire. La deuxième a consisté à proposer un modèle des contraintes binaires à l'aide de k-nn avec le noyau d'Epanechnikov. La troisième a été d'incorporer ces nouveaux modèles pour les utiliser dans $\tilde{h}$. Les résultats ont montré que l'utilisation de ces modèles pour les contraintes binaires permettait d'avoir de meilleurs ordonnancements. Les nouvelles versions ont toutes obtenues de meilleurs résultats que MADS par défaut. Il est donc recommandé d'incorporer ces modèles de contraintes binaires et ces traitements de contraintes binaires, non-relaxables et cachées dans NOMAD. La dernière contribution a été de vérifier l'impact d'une mise à l'échelle avec ces modèles de contraintes binaires. Les différences ont été peu significatives, mais il semble cependant que, même dans ce cadre, une mise à l'échelle soit parfaitement justifée d'un point de vue numérique.

7.2 Améliorations et travaux futurs

Des améliorations pourraient être entreprises en apprentissage sur les contraintes binaires pour avoir une meilleure compréhension de celles-ci. Notamment, d'autres méthodes de classification supervisée pourraient être testées, pour confirmer ou améliorer le choix expliqué en section 3.3.

Le chapitre 5 proposait de mettre les contraintes à l'échelle pour qu'elles aient globalement le même poids. Ceci leur a donné la même importance dans la suite de fonctions de violations de contraintes. Ce choix peut sembler arbitraire. Une sélection des variables les plus influentes a par exemple été utilisée dans un contexte de parallélisation en optimisation sans dérivées [4]. Ainsi, une méthode qui apprendrait dynamiquement l'importance de chacune des contraintes, en les hiérarchisant ou en les pondérant, pourrait être envisagée.

Enfin, les contraintes binaires ne donnant aucune nuance pour des points non-réalisables, celles qui sont relaxables pourraient avoir un traitement dans la barrière progressive en ne pénalisant pas par $\max(0, b(x))^2$ dans h (où b est une fonction à valeurs binaires), mais peut-être par $\max(0, \tilde{b}(x))^2$. En effet, un point serait moins pénalisé si l'on pense qu'il aurait de fortes chances d'être réalisable. Cela entrainerait deux difficultés notables. La première est que la fonction de violations de contraintes changerait en fonction des itérations. Cela constituait l'une des difficultés du chapitre 5, mais l'analyse de convergence semble moins claire dans ce cadre. De plus, l'utilisation d'un modèle dans une fonction de violations de contraintes ferait co-exister, dans la même fonction, des valeurs brutes de contraintes avec des modèles. Ceci pourrait créer des asymétries dans les différentes contraintes.

Enfin, le chapitre 6 a proposé de modéliser l'ensemble des contraintes non-relaxables par une seule contrainte binaire. Lorsque plusieurs contraintes non-relaxables sont identifiées, une nouvelle formulation du problème initial modélisant chaque contrainte non-relaxable par une contrainte binaire pourrait être choisie.

RÉFÉRENCES

[1] C. Audet et W. Hare, *Derivative-Free and Blackbox Optimization*, ser. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, 2017.

[2] C. Audet et J. E. Dennis, Jr., "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, n°. 1, p. 188–217, janv. 2006.

[3] C. Audet et J. Dennis, "A progressive barrier for derivative-free nonlinear programming," *SIAM Journal on Optimization*, vol. 20, n°. 1, p. 445–472, 2009. [En ligne]. Disponible : https://doi.org/10.1137/070692662

[4] N. Amaioua, "Modèles quadratiques et décomposition parallèle pour l'optimisation sans dérivées," Thèse de doctorat, École Polytechnique de Montréal, juin 2018. [En ligne]. Disponible : https://publications.polymtl.ca/3186/

[5] C. Audet, S. Le Digabel et M. Peyrega, "Linear equalities in blackbox optimization," *Computational Optimization and Applications*, vol. 61, n°. 1, p. 1–23, May 2015. [En ligne]. Disponible : https://doi.org/10.1007/s10589-014-9708-2

[6] L. A. Sarrazin-Mc Cann, "Opportunisme et ordonnancement en optimisation sans dérivées," Mémoire de maîtrise, École Polytechnique de Montréal, mai 2018. [En ligne]. Disponible : https://publications.polymtl.ca/3099/

[7] C. Audet, S. Le Digabel et C. Tribes, "Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization," *Optimization and Engineering*, vol. 17, n°. 2, p. 333–358, Jun 2016. [En ligne]. Disponible : https://doi.org/10.1007/s11081-015-9283-0

[8] A. Conn et S. Le Digabel, "Use of quadratic models with mesh-adaptive direct search for constrained black box optimization," *Optimization Methods and Software*, vol. 28, n°. 1, p. 139–158, 2013. [En ligne]. Disponible : http://dx.doi.org/10.1080/10556788.2011.623162

[9] S. Le Digabel et S. Wild, "A Taxonomy of Constraints in Simulation-Based Optimization," Les cahiers du GERAD, Rapport technique G-2015-57, 2015. [En ligne]. Disponible : http://www.optimization-online.org/DB_HTML/2015/05/4931.html

[10] T. D. Choi, O. J. Eslinger, C. T. Kelley, J. W. David et M. Etheridge, "Optimization of automotive valve train components with implicit filtering," *Optimization and Engineering*, vol. 1, n°. 1, p. 9–27, Jun 2000. [En ligne]. Disponible : https://doi.org/10.1023/A:1010071821464

[11] X. Chen et C. T. Kelley, “Optimization with hidden constraints and embedded monte carlo computations,” *Optimization and Engineering*, vol. 17, n°. 1, p. 157–175, Mar 2016. [En ligne]. Disponible : https://doi.org/10.1007/s11081-015-9302-1

[12] C. Poissant, “Exploitation d’une structure monotone en recherche directe pour l’optimisation de boîtes grises,” Mémoire de maîtrise, École Polytechnique de Montréal, février 2018. [En ligne]. Disponible : https://publications.polymtl.ca/3006/

[13] C. Audet, P. Côté, C. Poissant et C. Tribes, “Monotonic grey box optimization,” GERAD, HEC Montréal, Rapport technique Les Cahiers du GERAD G-2019-15, févr. 2019.

[14] C. Audet et C. Tribes, “Mesh-based Nelder–Mead algorithm for inequality constrained optimization,” *Computational Optimization and Applications*, June 2018. [En ligne]. Disponible : https://doi.org/10.1007/s10589-018-0016-0

[15] H. Abdulkarim et I. F Alshammari, “Comparison of algorithms for solving traveling salesman problem,” *International Journal of Engineering and Advanced Technology*, vol. ISSN, p. 2249 – 8958, 08 2015.

[16] A. Goder et V. Filkov, “Consensus clustering algorithms : Comparison and refinement,” dans *Proceedings of the Meeting on Algorithm Engineering & Expermiments*. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics, 2008, p. 109–117. [En ligne]. Disponible : http://dl.acm.org/citation.cfm?id=2791204.2791215

[17] J. A. Nelder et R. Mead, “A simplex method for function minimization,” *The Computer Journal*, vol. 7, n°. 4, p. 308–313, 1965. [En ligne]. Disponible : http://dx.doi.org/10.1093/comjnl/7.4.308

[18] M. J. D. Powell, “An efficient method for finding the minimum of a function of several variables without calculating derivatives,” *The Computer Journal*, vol. 7, n°. 2, p. 155, 1964. [En ligne]. Disponible : http://dx.doi.org/10.1093/comjnl/7.2.155

[19] R. Fletcher, “Function minimization without evaluating derivatives—a review,” *The Computer Journal*, vol. 8, n°. 1, p. 33–41, 1965.

[20] H. Ismkhan, “Black box optimization using evolutionary algorithm with novel selection and replacement strategies based on similarity between solutions,” *Applied Soft Computing*, vol. 64, p. 260 – 271, 2018. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S1568494617307214

[21] D. Golovin, G. Kochanski et J. E. Karro, “Black box optimization via a bayesian-optimized genetic algorithm,” dans *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017, to be submitted to Opt2017 Optimization for Machine Learning, at NIPS 2017.

[22] M. L. Garneau, “Modelling of a solar thermal power plant for benchmarking blackbox optimization solvers,” Mémoire de maîtrise, École Polytechnique de Montréal, déc. 2015. [En ligne]. Disponible : https://publications.polymtl.ca/1996/

[23] J.-P. Harvey et A. E. Gheribi, “Process simulation and control optimization of a blast furnace using classical thermodynamics combined to a direct search algorithm,” *Metallurgical and Materials Transactions B*, vol. 45, n°. 1, p. 307–327, Feb 2014. [En ligne]. Disponible : https://doi.org/10.1007/s11663-013-0004-9

[24] S. E. Skutnik et D. R. Davis, “Characterization of the non-uniqueness of used nuclear fuel burnup signatures through a mesh-adaptive direct search,” *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 817, p. 7 – 18, 2016. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0168900216001571

[25] S. S. S. Hosseini, A. Jafarnejad, A. H. Behrooz et A. H. Gandomi, “Combined heat and power economic dispatch by mesh adaptive direct search algorithm,” *Expert Systems with Applications*, vol. 38, n°. 6, p. 6556 – 6564, 2011. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0957417410013217

[26] Y. Ahn, C.-G. Lee, Y.-S. Jeong, Y.-J. Kim et S.-Y. Jung, “Optimal design of direct-driven pm wind generator using memetic algorithm coupled with fem,” 11 2009.

[27] A. Gheribi, S. Le Digabel, C. Audet et P. Chartrand, “Identifying optimal conditions for magnesium based alloy design using the Mesh Adaptive Direct Search algorithm,” *Thermochimica Acta*, vol. 559, n°. 0, p. 107–110, 2013. [En ligne]. Disponible : http://dx.doi.org/10.1016/j.tca.2013.02.004

[28] C. Audet, V. Béchard et S. Le Digabel, “Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search,” *Journal of Global Optimization*, vol. 41, n°. 2, p. 299–318, juin 2008. [En ligne]. Disponible : http://dx.doi.org/10.1007/s10898-007-9234-1

[29] A. Gheribi, C. Robelin, S. Le Digabel, C. Audet et A. Pelton, “Calculating all local minima on liquidus surfaces using the FactSage software and databases and the Mesh Adaptive Direct Search algorithm,” *The Journal of Chemical Thermodynamics*, vol. 43, n°. 9, p. 1323–1330, 2011. [En ligne]. Disponible : http://dx.doi.org/10.1016/j.jct.2011.03.021

[30] R. Hayes, F. Bertrand, C. Audet et S. Kolaczkowski, “Catalytic combustion kinetics : Using a direct search algorithm to evaluate kinetic parameters from light-off curves,” *The Canadian Journal of Chemical Engineering*, vol. 81, n°. 6, p. 1192–1199, 2003. [En ligne]. Disponible : http://pubs.nrc-cnrc.gc.ca/cjche/ch811192-6.html

[31] C. Parson, J. Chrissis, A. Palazotto et R. O'Hara, "Direct search optimization of a flapping micro air vehicle wing using fea characterization of the manduca sexta forewing," *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 01 2013.

[32] M. Pourbagian et W. Habashi, "CFD-Based Optimization of Electro-Thermal Wing Ice Protection Systems in De-Icing Mode," dans *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013*, 01 2013.

[33] R. Torres, J. Chaptal, C. Bes et J. Hiriart-urruty, "Multi-Objective Clean Take-Off Flight Paths for Civil Aircraft," dans *9th AIAA Aviation Technology, Integration, and Operations Conference, Hilton Head, South Carolina (USA), 21-23 September*, 2009.

[34] A. Das, Z. Gao, P. P Menon, J. Hardman et D. G Bates, "A systems engineering approach to validation of a pulmonary physiology simulator for clinical applications," *Journal of the Royal Society, Interface / the Royal Society*, vol. 8, p. 44–55, 01 2011.

[35] A. Das, P. P. Menon, J. Hardman et D. G. Bates, "Optimization of mechanical ventilator settings for pulmonary disease states," *IEEE transactions on bio-medical engineering*, vol. 60, 01 2013.

[36] A. Marsden, J. Feinstein et C. Taylor, "A computational framework for derivative-free optimization of cardiovascular geometries," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, n°. 21–24, p. 1890–1905, 2008. [En ligne]. Disponible : http://dx.doi.org/10.1016/j.cma.2007.12.009

[37] P. Evans, A. Castellazzi, M. Johnson, H. Lu et C. Bailey, "The optimization of thermal performance in power electronics modules," 05 2014, p. 734–739.

[38] L. S. Matott, K. Leung et J. Sim, "Application of MATLAB and python optimizers to two case studies involving groundwater flow and contaminant transport modeling," *Computers & Geosciences*, vol. 37, n°. 11, p. 1894–1899, 2011.

[39] O. Dubé, D. Dubé, J. Chaouki et F. Bertrand, "Optimization of detector positioning in the radioactive particle tracking technique," *Applied radiation and isotopes : including data, instrumentation and methods for use in agriculture, industry and medicine*, vol. 89C, p. 109–124, 02 2014.

[40] V. Ayma, P. Achanccaray Diaz, R. Feitosa, P. Nigri Happ, G. Costa, T. Klinger et C. Heipke, "Metaheuristics for supervised parameter tuning of multiresolution segmentation," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, p. 1364 – 1368, 09 2016.

[41] C. Audet, "Tuning Runge-Kutta parameters on a family of ordinary differential equations," *International Journal of Mathematical Modelling and Numerical Optimisation*,

vol. 8, n°. 3, p. 277–286, 2018. [En ligne]. Disponible : http://www.inderscience.com/info/inarticletoc.php?jcode=ijmmno&year=2018&vol=8&issue=3#issue

[42] E. Fermi et N. Metropolis, "Numerical solution of a minimum problem," Report LA-1492, nov. 1952.

[43] T. G. Kolda, R. M. Lewis et V. Torczon, "Optimization by direct search : New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, n°. 3, p. 385–482, 2003. [En ligne]. Disponible : https://doi.org/10.1137/S003614450242889

[44] V. Torczon, "On the convergence of pattern search algorithms," *SIAM J. on Optimization*, p. 1–25, 1997. [En ligne]. Disponible : http://dx.doi.org/10.1137/S1052623493250780

[45] C. Audet, "A short proof on the cardinality of maximal positive bases," *Optimization Letters*, vol. 5, n°. 1, p. 191–194, 2011. [En ligne]. Disponible : http://dx.doi.org/10.1007/s11590-010-0229-3

[46] J. Chen et P. Z. G. Qian, "Latin hypercube designs with controlled correlations and multi-dimensional stratification," *Biometrika*, vol. 101, n°. 2, p. 319–332, 02 2014. [En ligne]. Disponible : https://dx.doi.org/10.1093/biomet/ast062

[47] N. Mladenović et P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, n°. 11, p. 1097–1100, nov. 1997. [En ligne]. Disponible : http://dx.doi.org/10.1016/S0305-0548(97)00031-2

[48] F. Clarke, *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics, 1990. [En ligne]. Disponible : http://epubs.siam.org/doi/abs/10.1137/1.9781611971309

[49] C. Audet et J. E. Dennis, Jr., "Analysis of generalized pattern searches," *SIAM Journal on Optimization*, n°. 3, p. 889–903, janv. 2003.

[50] T. Kolda, R. Lewis et V. Torczon, "Optimization by direct search : New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, n°. 3, p. 385–482, 2003. [En ligne]. Disponible : https://doi.org/10.1137/S003614450242889

[51] C. Audet, S. Le Digabel et C. Tribes, "NOMAD user guide," Les cahiers du GERAD, Rapport technique G-2009-37, 2009. [En ligne]. Disponible : https://www.gerad.ca/nomad/Downloads/user_guide.pdf

[52] I. D. Coope et C. J. Price, "Frame based methods for unconstrained optimization," *Journal of Optimization Theory and Applications*, vol. 107, n°. 2, p. 261–274, Nov 2000. [En ligne]. Disponible : https://doi.org/10.1023/A:1026429319405

[53] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon et M. W. Trosset, “A rigorous framework for optimization of expensive functions by surrogates,” *Structural optimization*, vol. 17, n°. 1, p. 1–13, 1999. [En ligne]. Disponible : http://dx.doi.org/10.1007/BF01197708

[54] R. Tournemenne, J.-F. Petiot, B. Talgorn et M. Kokkolaras, “Brass instruments design using physics-based sound simulation models and surrogate-assisted derivative-free optimization,” *Journal of Mechanical Design*, vol. 139, n°. 4, p. –, avr. 2017.

[55] B. Talgorn, S. Le Digabel et M. Kokkolaras, “Problem Formulations for Simulation-Based Design Optimization Using Statistical Surrogates and Direct Search,” dans *Proceedings of The ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, vol. 2B : 40th Design Automation Conference, n°. DETC2014-34778. Buffalo, New York, USA : ASME, 2014, p. 12. [En ligne]. Disponible : http://dx.doi.org/10.1115/DETC2014-34778

[56] C. Audet, M. Kokkolaras, S. Le Digabel et B. Talgorn, “Order-based error for managing ensembles of surrogates in mesh adaptive direct search,” *Journal of Global Optimization*, vol. 70, p. 645–675, oct. 2018.

[57] C. Audet, A. R. Conn, S. Le Digabel et M. Peyrega, “A progressive barrier derivative-free trust-region algorithm for constrained optimization,” *Computational Optimization and Applications*, vol. 71, n°. 2, p. 307–329, Nov 2018. [En ligne]. Disponible : https://doi.org/10.1007/s10589-018-0020-4

[58] C. Audet, J. E. Dennis et S. Le Digabel, “Globalization strategies for mesh adaptive direct search,” *Computational Optimization and Applications*, vol. 46, n°. 2, p. 193–215, Jun 2010. [En ligne]. Disponible : https://doi.org/10.1007/s10589-009-9266-1

[59] R. T. Rockafellar, *Generalized Subgradients in Mathematical Programming*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1983, p. 368–390. [En ligne]. Disponible : https://doi.org/10.1007/978-3-642-68874-4_15

[60] M. A. Abramson, C. Audet, J. E. Dennis, Jr. et S. Le Digabel, “Orthomads : A deterministic mads instance with orthogonal directions,” *SIAM Journal on Optimization*, vol. 20, n°. 2, p. 948–966, janv. 2009.

[61] C. Audet, S. Le Digabel, C. Tribes et V. Rochon Montplaisir, “The NOMAD project,” Software available at https://www.gerad.ca/nomad/. [En ligne]. Disponible : https://www.gerad.ca/nomad/

[62] S. Le Digabel, “Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS Algorithm,” *ACM Trans. Math. Softw.*, vol. 37, n°. 4, p. 44 :1–44 :15, févr. 2011. [En ligne]. Disponible : http://doi.acm.org/10.1145/1916461.1916468

[63] E. Dolan et J. Moré, “Benchmarking optimization software with performance profiles,” *Mathematical Programming*, vol. 91, n°. 2, p. 201–213, 2002. [En ligne]. Disponible : http://dx.doi.org/10.1007/s101070100263

[64] J. J. Moré et S. M. Wild, “Benchmarking derivative-free optimization algorithms,” *SIAM Journal on Optimization*, vol. 20, n°. 1, p. 172–191, 2009. [En ligne]. Disponible : https://doi.org/10.1137/080724083

[65] V. Beiranvand, W. Hare et Y. Lucet, “Best practices for comparing optimization algorithms,” *Optimization and Engineering*, vol. 18, n°. 4, p. 815–848, Dec 2017. [En ligne]. Disponible : https://doi.org/10.1007/s11081-017-9366-1

[66] Y. Le Cun, L. D. Jackel, H. A. Eduard, N. Bottou, C. Cartes, J. S. Denker, H. Drucker, E. Sackinger, P. Simard et V. Vapnik, “Learning algorithms for classification : A comparison on handwritten digit recognition,” dans *Neural Networks : The Statistical Mechanics Perspective*. World Scientific, 1995, p. 261–276.

[67] N. Razali, A. Mustapha, F. Ahmad Yatim et R. Ab Aziz, “Predicting football matches results using bayesian networks for english premier league (epl),” *IOP Conference Series : Materials Science and Engineering*, vol. 226, p. 012099, 08 2017.

[68] S. A Czepiel, “Maximum likelihood estimation of logistic regression models : Theory and implementation,” 10 2017.

[69] G. Baudat et F. Anouar, “Generalized discriminant analysis using a kernel approach,” *Neural Comput.*, vol. 12, n°. 10, p. 2385–2404, oct. 2000. [En ligne]. Disponible : http://dx.doi.org/10.1162/089976600300014980

[70] S. Mika, G. Ratsch, J. Weston, B. Scholkopf et K. R. Mullers, “Fisher discriminant analysis with kernels,” dans *Neural Networks for Signal Processing IX : Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, Aug 1999, p. 41–48.

[71] H. Parvin, H. Alizadeh et B. Minaei, “Mknn : Modified k-nearest neighbor,” *Lecture Notes in Engineering and Computer Science*, vol. 2173, 10 2008.

[72] R. O. Duda, P. E. Hart et D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[73] S. Arlot et A. Celisse, “A survey of cross-validation procedures for model selection,” *Statist. Surv.*, vol. 4, p. 40–79, 2010. [En ligne]. Disponible : https://doi.org/10.1214/09-SS054

[74] N. Bhatia et Vandana, “Survey of nearest neighbor techniques,” *CoRR*, vol. abs/1007.0085, 2010. [En ligne]. Disponible : http://arxiv.org/abs/1007.0085

[75] T. Hastie, R. Tibshirani et J. Friedman, *The elements of statistical learning : data mining, inference and prediction*, 2e éd. Springer, 2009. [En ligne]. Disponible : http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[76] J. Gou, L. Du, Y. Zhang et T. Xiong, "A new distance-weighted k -nearest neighbor classifier," *J. Inf. Comput. Sci.*, vol. 9, 11 2011.

[77] P. Li, J. Gou et H. Yang, "The distance-weighted k-nearest centroid neighbor classification," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 8, p. 611–622, 01 2017.

[78] A. Z. Zambom et R. Dias, "A review of kernel density estimation with applications to econometrics," *International Econometric Review (IER)*, vol. 5, n°. 1, p. 20–42, 2013. [En ligne]. Disponible : http://EconPapers.repec.org/RePEc:erh:journl:v:5:y:2013:i:1:p:20-42

[79] T. Ledel, "Kernel density estimation : Theory and application in discriminant analysis," *Austrian Journal of Statistics*, vol. 33, n°. 2, p. 267–279, 2004.

[80] M. C. Jones, J. S. Marron et S. J. Sheather, "A brief survey of bandwidth selection for density estimation," *Journal of the American Statistical Association*, vol. 91, n°. 433, p. 401–407, 1996. [En ligne]. Disponible : https://www.tandfonline.com/doi/abs/10.1080/01621459.1996.10476701

[81] B. Hansen, "Exact mean integrated squared error of higher order kernel estimators," *Econometric Theory*, vol. 21, n°. 06, p. 1031–1057, 2005.

[82] A. Tharwat, T. Gaber, A. Ibrahim et A. E. Hassanien, "Linear discriminant analysis : A detailed tutorial," *AI Communications*, vol. 30, p. 169–190,, 05 2017.

[83] M. Pohar Perme, M. Blas et S. Turk, "Comparison of logistic regression and linear discriminant analysis : a simulation study," *Metodološki zvezki*, vol. 1, n°. 1, p. 143–161, 2004. [En ligne]. Disponible : http://www.dlib.si/details/URN:NBN:SI:doc-6C3HH0D0

[84] S. Mika, G. Rätsch, J. Weston, B. Schölkopf et K.-R. Müller, "Fisher discriminant analysis with kernels," 1999.

[85] C.-Y. J. Peng, K. L. Lee et G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *The Journal of Educational Research*, vol. 96, n°. 1, p. 3–14, 2002. [En ligne]. Disponible : https://doi.org/10.1080/00220670209598786

[86] S. A Czepiel, "Maximum likelihood estimation of logistic regression models : Theory and implementation," 2002.

[87] C. Zorn, "A solution to separation in binary response models," *Political Analysis*, vol. 13, n°. 02, p. 157–170, 2005. [En ligne]. Disponible : https://EconPapers.repec.org/RePEc:cup:polals:v:13:y:2005:i:02:p:157-170_00

[88] C. Rainey, “Dealing with separation in logistic regression models,” *Political Analysis*, vol. 24, n°. 3, p. 339–355, 2016.

[89] E. Makalic et D. F. Schmidt, “Review of modern logistic regression methods with application to small and medium sample size problems,” dans *AI 2010 : Advances in Artificial Intelligence*, J. Li, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 213–222.

[90] J. Schmidhuber, “Deep learning in neural networks : An overview,” *CoRR*, vol. abs/1404.7828, 2014. [En ligne]. Disponible : http://arxiv.org/abs/1404.7828

[91] B. J. A. Kröse et P. P. van der Smagt, *An Introduction to Neural Networks*, 4e éd. Amsterdam, The Netherlands : The University of Amsterdam, 1991.

[92] S. Dodge et L. Karam, “Understanding how image quality affects deep neural networks,” dans *2016 8th International Conference on Quality of Multimedia Experience, QoMEX 2016*. United States : Institute of Electrical and Electronics Engineers Inc., 6 2016.

[93] C. Cortes et V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, n°. 3, p. 273–297, sept. 1995. [En ligne]. Disponible : http://dx.doi.org/10.1023/A:1022627411411

[94] J. M. Moguerza et A. Muñoz, “Support vector machines with applications,” *Statist. Sci.*, vol. 21, n°. 3, p. 322–336, 08 2006. [En ligne]. Disponible : https://doi.org/10.1214/088342306000000493

[95] B. Schölkopf, C. Burges et V. Vapnik, “Extracting support data for a given task,” dans *Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park*. AAAI Press, 1995, p. 252–257.

[96] E. Zanaty, “Support vector machines (svms) versus multilayer perception (mlp) in data classification,” *Egyptian Informatics Journal*, vol. 13, n°. 3, p. 177 – 183, 2012. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S1110866512000345

[97] W. Yu, T. Liu, R. Valdez, M. Gwinn et M. J. Khoury, “Application of support vector machine modeling for prediction of common diseases : the case of diabetes and pre-diabetes,” *BMC Medical Informatics and Decision Making*, vol. 10, n°. 1, p. 16, Mar 2010. [En ligne]. Disponible : https://doi.org/10.1186/1472-6947-10-16

[98] B. Schölkopf, A. Smola, A. Smola et A. J Smola, “Support vector machines and kernel algorithms,” *Encyclopedia of Biostatistics, 5328-5335 (2005)*, 04 2002.

[99] J. Platt, “Sequential minimal optimization : A fast algorithm for training support vector machines,” *Advances in Kernel Methods-Support Vector Learning*, vol. 208, 07 1998.

[100] S. Amari et S. Wu, “Improving support vector machine classifiers by modifying kernel functions,” *Neural Networks*, vol. 12, nº. 6, p. 783–789, 1999.

[101] Z. Zhang, “Customizing kernels in support vector machines,” Mémoire de maîtrise, University of Waterloo, 2007.

[102] H. Han et X. Jiang, “Overcome support vector machine diagnosis overfitting,” *Cancer Informatics*, p. 145–158, 12 2014. [En ligne]. Disponible : www.la-press.com/overcome-support-vector-machine-diagnosis-overfitting-article-a4565

[103] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola et V. Vapnik, “Support vector regression machines,” dans *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan et T. Petsche, édit. MIT Press, 1997, p. 155–161. [En ligne]. Disponible : http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf

[104] M. Denil, D. Matheson et N. D. Freitas, “Narrowing the gap : Random forests in theory and in practice,” dans *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing et T. Jebara, édit., vol. 32, nº. 1. Bejing, China : PMLR, 22–24 Jun 2014, p. 665–673. [En ligne]. Disponible : http://proceedings.mlr.press/v32/denil14.html

[105] G. Louppe, “Understanding random forests,” Thèse de doctorat, University de Liège, Juillet 2014.

[106] A. Pretorius, S. Bierman et S. J. Steel, “A meta-analysis of research in random forests for classification,” dans *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, Nov 2016, p. 1–6.

[107] G. Biau, L. Devroye et G. Lugosi, “Consistency of random forests and other averaging classifiers,” *Journal of Machine Learning Research*, vol. 9, p. 2015–2033, 2008. [En ligne]. Disponible : https://dl.acm.org/citation.cfm?id=1442799

[108] C. Tang, D. Garreau et U. von Luxburg, “When do random forests fail ?” dans *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi et R. Garnett, édit. Curran Associates, Inc., 2018, p. 2983–2993. [En ligne]. Disponible : http://papers.nips.cc/paper/7562-when-do-random-forests-fail.pdf

[109] B. Talgorn, S. Le Digabel et M. Kokkolaras, “Statistical surrogate formulations for simulation-based design optimization,” *Journal of Mechanical Design*, vol. 137, nº. 2, p. 021 405/1–021 405/18, janv. 2015.

[110] C. Audet, A. Ianni, S. Le Digabel et C. Tribes, “Reducing the number of function evaluations in mesh adaptive direct search algorithms,” *SIAM Journal on Optimization*, vol. 24, n°. 2, p. 621–642, 2014. [En ligne]. Disponible : http://dx.doi.org/10.1137/120895056

[111] N. Gould, D. Orban et P. Toint, “CUTEr (and SifDec) : A constrained and unconstrained testing environment, revisited,” *ACM Transactions on Mathematical Software*, vol. 29, n°. 4, p. 373–394, 2003. [En ligne]. Disponible : http://dx.doi.org/10.1145/962437.962439

[112] A.-R. Hedar, “Global Optimization Test Problems,” http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm, (last accessed on 2017-10-20). [En ligne]. Disponible : http://goo.gl/0vxil

[113] L. Lukšan et J. Vlček, “Test problems for nonsmooth unconstrained and linearly constrained optimization,” ICS AS CR, Rapport technique V-798, 2000. [En ligne]. Disponible : http://www.cs.cas.cz/ics/reports/v798-00.ps

[114] C. Audet, J. Dennis, Jr. et S. Le Digabel, “Parallel Space Decomposition of the Mesh Adaptive Direct Search Algorithm,” *SIAM Journal on Optimization*, vol. 19, n°. 3, p. 1150–1170, 2008. [En ligne]. Disponible : http://dx.doi.org/10.1137/070707518

[115] W. Hock et K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, ser. Lecture Notes in Economics and Mathematical Systems. Berlin, Germany : Springer, 1981, vol. 187.

[116] C. Tribes, J.-F. Dubé et J.-Y. Trépanier, “Decomposition of multidisciplinary optimization problems : formulations and application to a simplified wing design,” *Engineering Optimization*, vol. 37, n°. 8, p. 775–796, 2005. [En ligne]. Disponible : http://dx.doi.org/10.1080/03052150500289305

[117] F. Pigache, F. Messine et B. Nogarede, “Optimal design of piezoelectric transformers : A rational approach based on an analytical model and a deterministic global optimization,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 54, n°. 7, p. 1293–1302, July 2007.

[118] L. C. Cagnina, S. Esquivel et C. A. Coello Coello, “Solving engineering optimization problems with the simple constrained particle swarm optimizer,” *Informatica (Slovenia)*, vol. 32, p. 319–326, 01 2008.

[119] A. H. Gandomi, X.-S. Yang et A. H. Alavi, “Mixed variable structural optimization using firefly algorithm,” *Computers & Structures*, vol. 89, n°. 23, p. 2325 – 2336, 2011. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0045794911002185

[120] R. G. Regis, “Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points,” *Engineering Optimization*, vol. 46, n°. 2, p. 218–243, 2014. [En ligne]. Disponible : https://doi.org/10.1080/0305215X.2013.765000

[121] J. Na, Y. Lim et C. Han, “A modified direct algorithm for hidden constraints in an lng process optimization,” *Energy*, vol. 126, p. 488 – 500, 2017. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0360544217304164