UNIVERSITÉ DE MONTRÉAL

OPPORTUNISME ET ORDONNANCEMENT EN OPTIMISATION SANS-DÉRIVÉES

LOÏC ANTHONY SARRAZIN-MC CANN
DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AVRIL 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

OPPORTUNISME ET ORDONNANCEMENT EN OPTIMISATION SANS-DÉRIVÉES

présenté par : SARRAZIN-MC CANN Loïc Anthony
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. NOM Prénom, Doct., président
Mme NOM Prénom, Ph. D., membre et directrice de recherche
M. NOM Prénom, Ph. D., membre

DÉDICACE

. . .

REMERCIEMENTS

Texte.

RÉSUMÉ

Le résumé est un bref exposé du sujet traité, des objectifs visés, des hypothèses émises, des méthodes expérimentales utilisées et de l'analyse des résultats obtenus. On y présente également les principales conclusions de la recherche ainsi que ses applications éventuelles. En général, un résumé ne dépasse pas quatre pages.

Le résumé doit donner une idée exacte du contenu du mémoire ou de la thèse. Ce ne peut pas être une simple énumération des parties du document, car il doit faire ressortir l'originalité de la recherche, son aspect créatif et sa contribution au développement de la technologie ou à l'avancement des connaissances en génie et en sciences appliquées. Un résumé ne doit jamais comporter de références ou de figures.

ABSTRACT

Written in English, the abstract is a brief summary similar to the previous section (Résumé). However, this section is not a word for word translation of the French.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

CS	Coordinate Search
GPS	Generalized Pattern Search
MADS	Mesh Adaptive Direct Search
GSS	Generating Set Search
imfil	Implicit Filtering
BBO	Blackbox optimisation
DFO	Derivative-Free Optimization

LISTE DES SYMBOLES

B	Matrice du modèle quadratique
C^k	Matrice génératrice.
c	Contrainte quantifiable
D	Un ensemble générateur positif.
d	Direction de recherche
d_s	Profil de donnée du solveur s.
e_i	Direction coordonnée.
F^k	Cadre à l'itération k.
f	Fonction objectif.
f_Ω	Fonction objectif sous barrière extrême.
G	Matrice de base.
H^k	Ensemble des directions supplémentaires pour la SEARCH.
h	Quantification de la violation de contraintes
k	Compteur d'itérations.
L^k	Matrice de directions complémentaires à l'itération k.
l	Borne inférieur
M^k	Maillage à l'itération k.
n	Dimension du problème.
P^k	Ensemble des points de sonde à l'itération k.
P	Ensemble de problèmes.
p	Problème
S^k	Ensemble des points de sonde à l'itération k.
S	Ensemble de solveurs
s	Solveur
u	Borne supérieure
x^k	Centre de sonde à l'itération k.
β	Borne sur la longueur d'une direction.
Γ^k	Matrice génératrice de l'espace.
Δ^k	Longueur du cadre à l'itération k
δ^k	Longueur du pas à l'itération k
ϵ_{stop}	Critère d'arrêt de l'algorithme
$\kappa(D^k)$	Mesure cosinus de l'ensemble D^k.
λ^k	Paramètre de contraction.

ξ	Élément de la base naturelle.
$\rho(\delta^k)$	Fonction de force.
ρ_s	Profil de performance du solveur s.
τ	Paramètre d'ajustement du maillage.
ϕ^k	Paramètre d'expansion
Ω	Ensemble réalisable de solution

CHAPITRE 1 INTRODUCTION

1.1 Optimisation de boîtes noires

L'optimisation est l'étude de l'obtention d'un minimum ou d'un maximum pour un problème donné. Les problèmes sont définis par leur vecteur de variables x, la fonction objectif sous-jacente $f(x)$ ainsi que leur ensemble réalisable Ω. On cherche alors à résoudre :

$$\min_x \{f(x) \ : \ x \ \in \ \Omega\}. \tag{1.1}$$

Ce problème peut prendre plusieurs formes, et plusieurs disciplines de l'optimisation existent en conséquent. Les problèmes peuvent être tels que x est composé de variable continues, entières ou une combinaison des deux, pour lesquels existent l'optimisation continue, en nombres entiers et mixte. Une grande quantité de spécificités sur les variables, la fonction objectif est les contraintes mènent à une panoplie de sous-disciplines en optimisation.
Les principaux algorithmes d'optimisation exploitent la structure même du problème, telle la connaissance des dérivées du premier et du second degré pour la résolution. On pense ici à la méthode du simplexe pour l'optimisation linéaire [42], ou encore de les méthodes de Newton ou de Quasi-Newton pour l'optimisation non-linéaire [44].
Dans le cas traité dans cet ouvrage, on étudie un certain type de problème dit **boîte noire**. Les boîtes noires sont caractérisées par leur fonctionnement ; elles prennent en entrée un vecteur de variables, et en ressort un résultat. Les fonctions sous-jacentes sont inconnues, trop complexes ou trop instables pour en retirer une forme analytique. On peut résumer une boîte noire à un problème complexe instable, demandant beaucoup de ressources à résoudre, dont les optima locaux sont fréquents et dont les dérivées sont inconnues, difficilement obtenables ou même inexistante. Un problème de boîte noire est en réalité l'optimisation d'une fonction qui est un boîte noire, et dont les contraintes sont aussi évaluées par des boîtes noires.
La discipline de l'optimisation qui se concentre sur la résolution de problèmes dont la structure de la fonction objectif et des fonctions contraintes ne peux être exploitée se nomme l'optimisation de boîte noire (*BBO - Blackbox Optimization*). On distingue de cette discipline l'optimisation sans dérivées (*DFO - Derivative-free optimization*), dans laquelle on étudie la résolution de problème d'optimisation en n'utilisant aucune valeurs de dérivées de la fonction [8, 10] Ainsi, les outils de *DFO* utilisent seulement les valeurs de la fonction objectif calculées afin de bâtir des modèles qui seront eux-mêmes utilisés dans le processus d'optimisation. Contrairement à la *BBO*, la *DFO* ne sous-entends pas l'inexistence des déri-

vées de la fonction à optimiser, seulement qu'elles ne sont peut-être pas appropriées comme outil pour la résolution du problème.
Conn, Scheinberg et Vincente [17] regroupent les méthodes de *DFO* en deux familles : les méthodes de recherche directe et les méthodes de régions de confiance. Les méthodes de recherche directe se résument à ce que d'autres nomment les méthodes d'échantillonnage (sampling methods) [33] : l'optimisation est contrôlée par l'évaluation de $f(x)$ dans un ensemble de point, et c'est le résultat de ces évaluations qui détermine le prochain ensemble de point, jusqu'à la convergence. Les méthodes de régions de confiance, quant à elles, utilisent des modèles pour déterminer des points candidats, évaluent ces points et mettent à jour les modèles avec les résultats de ces points pour ensuite trouvé un autre point candidat et répéter ce processus jusqu'à la convergence.

1.2 Définition de la problématique

Les méthodes de recherche directe diffèrent des méthodes de régions de confiance de par leur structure. Plusieurs définitions existent pour les méthodes de recherche directe. Hooke and Jeeves [30] proposent en 1961 une définition qui éclaircie un peu l'idée derrière les méthodes.

> On utilise l'expression «recherche directe» pour décrire l'examination séquentielle de solutions candidates impliquant la comparaison de chaque candidat à la meilleure obtenue précédemment, avec une stratégie pour déterminer les prochains candidats [...].

Depuis l'avènement de problèmes dont les dérivées sont pratiquement inexistantes ou incalculables, les méthodes de recherche directe sont devenues au cœur des intérêt de plusieurs chercheurs. Cependant, puisque les évaluations sont couteuses, il y a la naissance d'un besoin de développer ces méthodes de façon à conserver leur fondements théoriques de convergence tout en ayant recours à un minimum d'évaluations de la fonction objectif. Par exemple, Audet et al. [9] ont travaillé à la génération d'un espace générateur avec le moins de directions possible pour ainsi réduire le nombre de directions nécessaires à chaque itération. À chaque itération, un algorithme de recherche directe possède une liste d'au moins un point candidat, tandis qu'un algorithme de régions de confiance n'en a assurément qu'un seul. Certaines méthode de recherche directe sont cependant similaires aux méthodes de régions de confiance en ce sens, tel que l'algorithme de Nelder-Meads [43], qui est classée comme une méthode de recherche directe simpliciale par Conn, Scheinberg et Vincente [17].
Puisque chaque évaluation de la fonction objectif est très couteuse en ressources, on en vient à se demander l'évaluation de chaque point dans la liste générée à chaque itération est nécessaire au bon déroulement de l'algorithme. Torczon [47] démontre qu'une famille d'algorithme

de recherche directe converge vers un optimum sans avoir à évaluer tous les points dans une liste générée à une itération donnée. Ainsi, la convergence n'est pas assurée par la qualité de la direction de descente utilisée à chaque itération $f(x^k+\delta^k d) = \min(f(x^k+\delta^k d : d \in D))$, mais par sa propriété de répondre à la définition de décroissance simple $f(x^k+\delta^k d) - f(x^k) < 0$, avec δ^k la longueur du pas et $d \in D^k$ la direction de descente choisie dans l'ensemble des directions D, pour l'itération k. On en vient à l'élaboration d'une problématique qui justifie le projet. Pour un algorithme possédant à l'itération k une solution courante x_k et un liste de candidat P^k, on cherche à déterminer si la découverte d'un nouveau meilleur point dans cette liste devrait entrainer un passage prématuré à l'itération suivante de l'algorithme.

1.3 Objectif de la recherche

Le long de ce mémoire, l'interruption prématurée d'une étape nécessaire à la convergence d'un algorithme sera nommée *stratégie opportuniste* ou *opportunisme*, de l'expression pour référer à cette nomenclature introduite par Coope et Price [18] et reprise par Audet et Dennis [4]. La question restant très vague, on devra définir plusieurs aspects afin d'en venir à une quantification de l'impact de l'opportunisme sur le déroulement d'algorithmes de *DFO*.
En premier lieu, la structure de l'algorithme doit admettre qu'au cours d'une itération il existe une liste de point à évaluer séquentiellement avant la détermination d'une autre liste de points. Ainsi, appartenir à la famille des méthodes de recherche directe n'est pas un critère suffisant compte tenu que l'algorithme de Nelder Meads y figure, malgré qu'il évalue un point à la fois. Cependant, d'autres famille d'algorithmes ouvrent la porte à l'opportunisme par leur fabrication. Notons ici les méthodes de recherche linéaire basée sur les dérivées simplexe, telles que baptisées par Conn, Scheinberg and Vincente [17], qui ne concordent pas exactement à la définition de méthodes de recherche directe. Un objectif premier de cette recherche est d'identifier les algorithmes laissant place à l'implémentation de l'opportunisme en eux.
De l'utilisation de l'opportunisme apparait une autre problématique au coeur de cette recherche, c'est à dire la stratégie utilisée pour ordonner les points candidats dans la liste. Si on permet une interruption de la séquence d'évaluation des points sur la liste lors de l'obtention d'un meilleur point, alors l'ordre dans laquelle apparaît ces points sera au cœur du comportement de l'algorithme. On nommera la règle définissant l'ordre des points dans la liste la *stratégie d'ordonnancement*. Le second objectif de cette recherche est d'identifier les stratégies d'ordonnancement qui pourraient avoir un impact sur la performance de la stratégie opportuniste.
Ayant en main un éventail d'algorithmes et de stratégies d'ordonnancement, on pourra procé-

der à l'objectif principal de la recherche, soit la quantification de la performance de l'utilisation de la stratégie opportuniste dans les algorithmes, ainsi que la comparaison des différentes stratégies d'ordonnancement identifiées. On cherche ainsi à déterminer les cas dans lesquels l'opportunisme est une stratégie envisageable, nécessaire, ou contraignante. On cherche aussi à caractériser les différentes stratégies d'ordonnancement en fonction du type de problème à résoudre, des spécificités algorithmiques ou en fonction de la précision demandée à l'algorithme.

1.4 Plan du mémoire

CHAPITRE 2 ALGORITHMES D'OPTIMISATION SANS-DÉRIVÉES

2.1 Coordinate Search

Le premier algorithme à être abordé est aussi le plus intuitif pour l'optimisation sans contraintes, qu'on appelera *Coordinate Search* (CS), ou Recherche par coordonnée, parfois appelée *Compass Search* [34]. On attribue à Fermi et Metropolis [24] cette première méthode de recherche directe. Pour que leur modèle suive bien leur ensemble de données expérimentales sur la diffusion nucléaire, Fermi et Metropolis ont fait varier les paramètre théoriques de déphasage de leur fonction un à la fois avec un pas constant. Lorsque ni l'augmentation ni la diminution de l'un des paramètres améliorait la concordance avec les données expérimentales, la longueur du pas était diminuée de moitiée et le processus était recommencé. On continuait ainsi jusqu'à ce que le pas soit considéré suffisement petit. C'est ainsi qu'est née la méthode de la recherche par coordonnée. Booker et al. [13] proposent un cadre rigoureux aux algorithmes de recherche de motifs. On y stipule que les algorithmes doivent être divisés en deux étapes, soient la recherche qu'on notera SEARCH et la sonde qu'on notera POLL afin d'être conforme avec la littérature. La SEARCH consiste à l'implentation d'une méthode visant à explorer le domaine de la fonction sans égard à la détermination d'un optimum. Le POLL cherche à minimiser la fonction pour déterminer un optimum. Dans le cadre qui nous intéresse, le POLL serait l'entièreté du processus, tandis que la SEARCH n'y trouverait pas son correspondant. La description de l'algorithme CS décrite en vertu du cadre proposé par Booker et al. est fortement inspirée de celle de Audet et Hare, issue de [8].

Algorithme 1 Recherche par coordonnée (CS)

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x^0 le point de départ

0. Initialisation des paramètres :

$\delta^0 \in (0, \infty)$ la longueur du pas initial

$\epsilon_{\text{stop}} \in [0, \infty)$ le critère d'arrêt

$k \leftarrow 0$ le compteur d'itérations

1. POLL

if $f(t) < f(x^k)$ pour un $t \in P^k := \{x^k \pm \delta^k e_i : i = 1, 2, \ldots, n\}$ **then**

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

else

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

end if

2. Terminaison

if $\delta^k \geq \epsilon_{\text{stop}}$ **then**

$k \leftarrow k + 1$

go to 1.

else

stop

end if

À l'étape POLL de l'algorithme, on entend que la fonction $f(x)$ est évaluée à chaque élément de $t \in P^k$ avant de déterminer quel t deviendra le nouveau centre de sonde x^{k+1}. Cependant, si les évaluations sont effectuées en série, on aura une séquence de points à évaluer. La séquence proposée est celle de $2n$ mouvements dans les directions elementaires suivie de la détermination d'un nouveau centre de sonde si au moins une évaluation est un succès. C'est dans cette séquence que l'opportunisme pourra être introduit, afin de ne pas évaluer le reste de la liste si le test de décroissance simple est positif.

2.2 Generalized Pattern Search

Le deuxième algorithme présenté est la recherche par motifs généralisée. La premiere recherche par motifs en soit est élaboré par Hooke et Jeeves [30]. Ils nomment la recherche par motifs (*Pattern Search* ou PS) la routine de recherche directe visant à minimiser une fonction $f(x)$ avec leur algorithme. Cette routine est composée d'une série de mouvements autour d'un point qui peuvent être divisés en deux types, soit les mouvements exploratoires (*Exploratory Moves*) ou les mouvements destinés à la détermination d'un minimum, soit les

mouvements de motifs (*Pattern Moves*). Les mouvements exploratoires servent à la détermination du motifs, c'est à dire au comportement de la fonction $f(x^k)$ aux alentours d'un point x^k. Les mouvements de motifs sont ensuite effectués dans la direction que les mouvements exploratoires ont déterminé comme étant celle qui se dirige vers le minimum de la fonction. Hooke et Jeeves introduisent aussi la définition de succès et d'échec. Un succès est mouvement tel que la valeur de $f(x)$ au point est inférieure à la meilleure connue auparavant. Dans le cas contraire, on dira que c'est un échec.
Lors de l'élaboration de cette technique ils mentionnent que :

> Par soucis de simplicité, les mouvements exploiratoires sont choisis de façon simple, c'est à dire, à chaque mouvement seulement la valeur d'une unique coordonée est changée.

Pour ensuite affirmer que :

> Suivant un mouvement de motif fructueux, il est raisonnable de conduire une série de mouvements exploratoires et de tenter d'améliorer d'avantage les résultats.

De ces deux affirmations découlent les principes de bases de PS, c'est à dire une sucscession de mouvements exploratoires dans les direction coordonnées $e_i : i = \pm\{1, 2 \dots n\}$ et d'un mouvement de motifs dans la meilleure direction. Chaque succès de la recherche de motifs entraine que la prochaine serie de mouvements exploratoires sera effectuée autour de ce nouveau meilleur point. Lorsque la succession échoue, la longueur du pas est réduite afin de déterminer un nouveau motif. Pour préciser la méthode, un point initial x^0 doit être déterminé ainsi qu'une longueur de pas initiale δ^0 et un critère ϵ_{stop} pour lequel on jugera que le pas est devenu suffisement petit. De plus, on doit déterminer la méthode utilisée pour procéder à la réduction de la longueur du pas. Dans les mots de Hooke et Jeeves, le POLL serait l'ensemble de mouvements exploratoires accompagné de la mise à jour du centre de sonde, tandis que la SEARCH serait le mouvement de motif restant.
Torczon [47] amène une généralistation du PS de Hooke and Jeeves. Dans cet article, l'auteure approche la méthode d'un autre oeil. Le motif propre à l'itération P^k est issu de la multiplication de deux matrices GC^k, soient $G \in \mathbb{R}^{n\times n}$ la matrice de base et $C^k \in \mathbb{R}^{n\times p}, p > 2n$, la matrice generatrice. La matrice de base se doit d'être non-singulière et la matrice génératrice se doit d'être composée telle que $C^k = [\Gamma^k \ L^k]$, ou Γ^k est la concaténation d'une matrice de plein rang M et de son opposée. Le rôle de Γ^k est de générer l'espace $\mathbb{R}^n$, tandis que le rôle de la L^k est de complémenter cette derniere avec des directions supplémentaires ne servant pas à strictement à générer l'espace, rappelant la SEARCH. Ainsi, GC^k est une matrice qui génére l'espace $\mathbb{R}^n$ et qui se libère du cadre restreignant des directions unitaires proposé par Fermi et Metropolis et repris par Hooke et Jeeves. Armés de GC^k et d'une longueur de pas spécifique à une itération δ^k, on retrouve $\delta^k G c_i^k$, soit une généralisation de $\delta^k e_i$ présent dans

CS, mais pour lequel on a une colonne $c_i^k \in C^k$ perturbée par G remplaçant la direction elementaire.

Toujours selon Torczon, la définition de mouvements exploratoires est reprises pour être plus générale. Un mouvement exploratoire est en fait un vecteur issu du motif $s^k \in P^k$. L'auteure demande aussi que, si il existe un vecteur colonne $c_i^k \in P^k$ tel que $f(x^k + \delta^k c_i^k) < f(x^k)$, alors les mouvements exploratoires doivent produire un pas s^k tel que $f(x^k + s^k) < f(x^k)$.

L'algorithme s'inscrit alors en cinq étapes. Premièrement, à l'initialisation de l'algorithme, vient le calcul de la fonction $f(x)$ à l'itéré initial x^0 à l'itération 0. Deuxièmement, le calcul d'une direction s_k issue de la série de mouvements exploratoires. Troisièment, le calcul de $f(x^k + s^k)$. Ensuite vient la mise à jour $x^{k+1} = x^k + s^k$ si l'étape précédente est un succès, sinon $x^{k+1} = x^k$. Enfin, au besoin, on met à jour l'ensemble générateur C^k et la longueur du pas δ^k et on recommence le processus à partir de a deuxième étape jusqu'à ce que δ^k soit jugé suffisamment petit.

Afin de rester dans le cadre de Booker et al. [13], on dépeint l'algoritme en se rattachant aux concepts de POLL et de SEARCH. Les directions issues de la matrice supplémentaire $f(x^k + s^k), s^k \in GL^k$ introduite par Torczon [47] correspondent à une étape de SEARCH en vertu de sa fonction d'incorporer des directions supplémetaires à la recherche, tandis que les mouvements exploratoires visant à trouver un minimum autour du point de sonde $(f(x^k + s^k), s^k \in D\Gamma^k)$ correspondent à la section POLL. Cependant, on généralisera d'avantage $D\Gamma^k$ afin de le remplacer par un ensemble générateur positif [22], noté $D = GZ, Z \in \mathbb{R}^{n\times p}$, qui réponds aux exigence exactes énoncées dans [47] sans imposer que la taille de la matrice bornée supérieurement à $p = 2n$, mais bien tel que $n + 1 \leq p \leq 2n$, en concordance avec les propriétés d'une base positive. Cette représentation est fortement inspirée de celle fournie par Audet et Hare dans [8]. On dira de $S^k = GL^k$ qu'il est l'ensemble des directions supplémentaires L^k fournit à l'itération k soumit à une transformation par la matrice inversible $G \in \mathbb{R}^{n\times n}$.

Par ailleurs, il est à noter que les algorithmes de recherche directe énoncés dans [35, 47, 36, 37], qui sont des algorithmes déclinant de la famille d'algorithmes GPS, diffèrent à la formulation donnée dans ce document inspirée de Audet et Hare [8], notamment sur un aspect principal. La mise à jour de la longueur du pas se fait par deux paramètre différents, soient $\delta^{k+1} = \phi^k \delta^k, \phi^k \geq 1$ pour l'expansion du maillage dans le cas ou l'itération k est un succès et $\delta^{k+1} = \lambda^k \delta^k, \lambda^k \in (0, 1)$ dans le cas d'un échec. Dans la formulation présente on utilise un seul paramètre $\tau \in (0, 1)$ et son inverse τ^{-1} pour cette mise à jour. Cependant, cette altération ne contrevient pas aux requis de la démonstration de la convergence de [47] qui assurent que la formulation présente possède les même propriétés. La formulation présente aussi une différence notable avec celle fait par Booker et al. dans [13], c'est à dire l'admission

de la mise à jour de la longueur du pas pour une SEARCH fructueuse, alors que Booker et al. incorporent la mise à jour seulement dans un échec de POLL, à l'image de l'utilisation d'un facteur $\phi^k = 1$.

Algorithme 2 Recherche par motifs généralisée (GPS)

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x^0 le point de départ

0. Initialisation des paramètres :

$\delta^0 \in (0, \infty)$	la longueur du pas initial
$D = GZ$	un ensemble générateur positif
$\tau \in (0, 1) \cup \mathbb{Q}$	le paramètre d'ajustement du maillage
$\epsilon_{\text{stop}} \in [0, \infty)$	le critère d'arrêt
$k \leftarrow 0$	le compteur d'itérations

1. SEARCH

if $f(t) < f(x^k)$ pour un $t \in S^k$ **then**
  $x^{k+1} \leftarrow t$ et $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$
  go to 3
else
  go to 2.
end if

2. POLL

détermination d'un ensemble générateur positif $\mathbb{D}^k \subseteq \mathbb{D}$
if $f(t) < f(x^k)$ pour un $t \in P^k := \{x^k + \delta^k d : d \in D^k\}$ **then**
  $x^{k+1} \leftarrow t$ et $\delta^{k+1} \leftarrow \tau^{-1}\delta^k$
else
  $x^{k+1} \leftarrow t$ et $\delta^k \leftarrow \tau\delta^k$
end if

3. Terminaison

if $\delta^k \geq \epsilon_{\text{stop}}$ **then**
  $k \leftarrow k + 1$
  go to 1.
else
  stop
end if

L'opportunisme pourra être implémenté dans l'étape de POLL selon la même logique que dans CS, c'est à dire à même la liste des points $x^k + \delta^k d : d \in D^k$. Pour ce qui est de la SEARCH, l'implémentation de la stratégie est faisable mais sa performance n'en vient qu'à

dépendre de la pertinence de la méthode pour générer S^k.

2.3 Mesh Adaptive Direct Search

Le troisième algorithme présenté est dans les même lignée que CS et GPS. Il s'agit de la recherche direct sur maillage adaptif (*Mesh Adaptive Direct Search* ou MADS), formulé par Audet et Dennis [5]. Contrairement aux algorithmes précédents, MADS est un algorithme basé sur un maillage, c'est à dire que chaque évaluation de la fonction objectif se fera à un point sur une discrétisation de l'espace des variables. On dénotera $M^k := \{x^k + \delta^k Dy : y \in \mathbb{N}^p\} \subset \mathbb{R}^n$ le maillage à l'itération k sur lequel les itérés seront définis, généré par l'ensemble générateur positif $D = GZ$ et y les entiers naturels. Il s'agit ici d'un maillage de cardinalité infini. Dans leur formulation de Audet et Hate reprise ici, les algorithmes CS et GPS sont aussi considéré comme étant basé sur un maillage, quoique leurs premières références [30, 47] n'en indique pas autant.

La grande distinction de MADS ne réside alors pas dans son utilisation d'un maillage comme structure, mais bien l'incorporation d'un cadre à son étape de sonde. On définit le cadre tel que $F^k := \{x \in M^k : ||x - x^k||_\infty \leq \Delta^k b\}$ à l'itération k, soit l'ensemble des points sur le maillage M^k pour lesquels la distance de Chebyshev à l'itéré courant est inférieur à une valeur $\Delta^k b$. Cette valeur est déterminée par $b = \max\{||d'||_\infty : d' \in \mathbb{D}\}$, soit la plus grande composante vectorielle présente dans tous les colonnes de la base positive $\mathbb{D}$ issue de l'ensemble générateur D, multipliée par un paramètre introduit dans MADS, soit le paramètre de longueur du cadre Δ^k.

La figure 2.1 montre une itération hypothétique de MADS pour laquelle $\Delta^k = \frac{1}{2}, \delta^k = \frac{1}{8}$ avec l'ensemble générateur étant $D = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$.

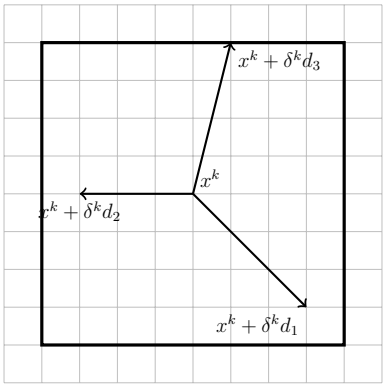

Figure 2.1 Ensemble générateur, maillage et cadre

Dans GPS, quoique l'algorithme décrit n'inclue pas de maillage formellement, les évaluations sont limitées aux directions de D mise à l'échelle avec la longueur du pas. Dans MADS, on pourra choisir n'importe quelle direction telle que $x^k + \delta^k d$,tant que l'ensemble des directions d forment un ensemble générateur positif D^k_Δ qui soit compris dans le cadre F^k. La mise à jour de la longueur du cadre se fait avec le paramètre d'ajustement du maillage qui, dans CS et GPS, servait pour la mise à jour de δ^k. Dans MADS, δ^k est mis à jour par défaut est d'utiliser $\delta^k = \min(\Delta^k, (\Delta^k)^2)$. Ainsi, on s'assure de respecter $\delta^k \leq \Delta^k$ et on augmente exponentiellement le nombre de directions possibles si $\Delta^k \leq 0$. Afin de rester dans le cadre de Booker et al. [13], on décrit l'algorithme en se rattachant aux concepts de POLL et SEARCH avec un étape SEARCH limitée à un ensemble S^k de points. La description de l'algorithme est fortement inspirée de celle de Audet et Hare, issue de [8].

Algorithme 3 Recherche par treillis adaptifs (MADS)

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x_0 le point de départ

0. Initialisation des paramètres :

$\Delta^0 \in (0, \infty)$	la longueur du cadre initial
$D = GZ$	un matrice génératrice positive
$\tau \in (0, 1) \cup \mathbb{Q}$	le paramètre d'ajustement du maillage
$\epsilon_{\text{stop}} \in [0, \infty)$	le critère d'arrêt
$k \leftarrow 0$	le compteur d'itérations

1. Mise à jour des paramètres

   $\delta^k \leftarrow \min(\Delta^k, (\Delta^k)^2)$

2. SEARCH

   if $f(t) < f(x^k)$ pour un $t \in S^k$ **then**

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

   go to 4

   else

   go to 3

   end if

3. POLL

   avec le cadre F^k de demi-coté Δ^k

   détermination d'un ensemble générateur positif $\mathbb{D}_\Delta^k \subset F^k$

   if $f(t) < f(x^k)$ pour un $t \in P^k := \{x^k + \delta^k d : d \in \mathbb{D}_\Delta^k\}$ **then**

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

   else

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

   end if

4. Terminaison

   if $\delta^k \geq \epsilon_{\text{stop}}$ **then**

   $k \leftarrow k + 1$

   go to 1.

   else

   stop

   end if

L'implémentation de la stratégie opportuniste sera faite à l'étape de POLL selon la même logique que pour CS et GPS.

2.4 Generating Set Search

Le quatrième algorithme est la recherche par ensemble générateur. Il est introduit par Kolda, Lewis et Torczon dans [34] sous le nom de *Generating Set Search* qu'on abrègera GSS. Il s'agit d'un algorithme très similaire à GPS mais qui incorpore certains aspects laissés de côté lors de la formulation de GPS et certains aspects nouveaux propre à GSS. À l'instar de GSS, un ensemble de direction à chaque itération est dédié à la SEARCH, soit H^k et un ensemble dédié à la POLL, soit D^k.

La différence principale entre GSS et GPS est le critère d'acceptation d'un itéré comme étant un succès. Pour les algorithmes précédents, chaque itération entrainant une simple diminution $f(x^k + \delta^k d) < f(x^k)$ était considérée comme un succès. Dans GSS, on permet que le critère d'acceptation soit resserré avec l'addition du terme $\rho(\delta^k)$ au test de diminution, soit que $f(x^k + \delta^k d) < f(x^k) + \rho(\delta^k)$. $\rho(\delta)$ sera appelée la fonction de force et doit satisfaire à une des deux définitions. Soit $\rho(\delta)$ est continue, $\rho(\delta) = o(\delta)$ lorsque $\delta \downarrow 0$ et que $\rho(\delta_1) < \rho(\delta_2)$ si $\delta_1 < \delta_2$, ou soit $\rho(\delta) \equiv 0$. Le premier cas impose une diminution suffisante de la fonction, rappelant le critère d'Armijo [2, 29]. Cet outil permet d'assurer la convergence globale théorique de l'algorithme en utilisant les résultats de convergence de Lucidi et Sciandrone [38] avec seulement une fonction objectif $f(x)$ bornée inférieurement. Le deuxième cas corresponds à une condition de diminution simple. Sous cette condition, les résultats de convergence globale sont assurés par les démonstrations faites sur les algorithmes basés sur treillis. Les conditions de convergences sont resserrés, de façon à ce que GSS convergera dans le cas où les paramètres fournies restreignent les points candidats à un maillage, et qu'aucune expansion de ce maillage n'est admise. [34, 18]. Il est à noter que CS et GPS sont des algorithmes de la famille de GSS, c'est à dire qu'une paramétrisation spécifique de GSS permets d'obtenir GPS ou CS.

Les auteurs spécifient deux autres grande différences en comparaison avec CS. Premièrement, on y fait valoir l'utilisation d'un ensemble générateur positif comme ensemble de directions de sonde, un aspect déjà incorporé dans notre définition de GPS. L'autre différence se situe dans la souplesse des paramètres de contraction τ^k et des paramètres d'expansion ϕ^k.
Plusieurs paramètres supplémentaires sont nécessaires tels que la mesure du cosinus, le plus petit angle entre deux directions de l'ensemble générateur $\kappa(G^k)$ qui empêche le choix de mauvaises directions, à l'instar de la mesure d'angle [29, 46] en recherche linéaire. On y fait mention aussi de la longueur des vecteurs de l'ensemble générateur étant bornées tel que $\beta_{\min} < \|d\| < \beta_{\max}, \beta_{\max} \geq \beta_{\min} > 0$.

Algorithme 4 Recherche par ensemble générateur (GSS)

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x^0 le point de départ

0. Initialisation des paramètres :

$\delta^0 \in (0, \infty)$	la longueur du pas initial
$\lambda_{\max} \in (0, 1) \cup \mathbb{R}$	le paramètre de contraction maximale de la longueur du pas
$\lambda^0 0 \in (0, 1) \cup \mathbb{R}, \phi^0$	les paramètre de contraction et d'expansion du pas
$\rho : [0, +\infty] \to \mathbb{R}$	une fonction continue tel que $\nabla(\rho(\delta)) < 0$ lorsque $\delta \to 0$ et $\frac{\rho(\delta)}{\delta} \to 0$ quand $\delta \downarrow 0$
$\beta_{\max} \geq \beta_{\min} > 0$	les bornes sur la longueur des vecteurs de G^k
$\epsilon_{\text{stop}} \in [0, \infty)$	le critère d'arrêt
$\kappa_{\min} \geq 0$	La mesure cosinus minimale d'un ensemble
$k \leftarrow 0$	le compteur d'itérations

1. SEARCH

détermination de $H^k = \{s \in \mathbb{R}^n,\ \beta_{\min} \leq \|s\|\}$

if $f(t) < f(x^k) - \rho(\delta^k)$ pour un $t \in S^k = \{x^k + \delta^k s, s \in H^k\}$ **then**

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

go to 3

else

go to 2.

end if

2. POLL

détermination de $D^k = \{s \in \mathbb{R}^n,\ \beta_{\min} \leq \|s\| \leq \beta_{\max},\ \kappa(H^k \cup D^k) \geq \kappa_{\min}\}$

un ensemble générateur

if $f(t) < f(x^k)$ pour un $t \in P^k := \{x^k + \delta^k d : d \in D^k\}$ **then**

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

else

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

end if

3. Terminaison

if $\delta^k \geq \epsilon_{\text{stop}}$ **then**

$k \leftarrow k + 1$

go to 1.

else

stop

end if

2.5 Implicit Filtering

Le dernier algorithme est celui du filtrage implicite (*Implicit Filtering* ou imfil), une méthode développée par Kelley [32, 33]. IMFIL se veut un algorithme d'optimisation sans-contraintes explicites hybride de recherche linéaire se basant sur le gradient simplex incorporant des particularités de CS. On définira le gradient simplex de IMFIL comme l'approximation du gradient en calculant la valeur de la fonction à chaque point d'un simplexe où il est possible de le faire.

$$\nabla_s f(x^k) = \frac{1}{h^k}\delta(f, x^k, V, h)V^\dagger$$

avec h le pas $V \in \mathbb{R}^{n\times k}$ une matrice représentant les directions du simplexe, $\delta(F, x, V, h^k)$ la matrice ayant $f(x+hv_j^1)-f(x)$ à sa j^e colonne, $\{v_1^j\} \in V$ et $V^\dagger$ son inverse de Moore-Penrose [26] et h^k le pas de l'itération. L'utilisation de la pseudo-inverse est justifiée par la nécessité de calculer des différences finies d'un seul côté si un point du simplexe n'est pas réalisable. L'idée fondamentale de IMFIL est de calculer le gradient simplex en un itéré courant x^k, avec des différences simples ou centrées, et d'utiliser ce gradient $\nabla_s f(x^k) \in \mathbb{R}^n$ comme direction pour effectuer une recherche linéaire à l'image d'une descente du gradient [45].

L'algorithme est défini comme sans-dérivées puisqu'il n'utilise pas le calcul de dérivées analytiques de la fonction. Cependant, contrairement aux étapes de POLL des algorithmes présentés précédemment, l'algorithme approxime la dérivée à l'aide de différences finies. Pour déterminer les valeurs de $f(x + hv_i^1)$ servant au calcul du gradient simplex, l'auteur introduit une nomenclature proche de la notre, soit le stencil poll, qu'on généralisera sonde du simplexe, qui est constituée de n à $2n$ évaluations de la boîte noire, dans les cas opposés où deux contraintes de bornes sont actives et aucune contrainte de borne n'est active, si le motif utilisé est une combinaison des vecteurs unitaires de $\mathbb{R}^n$. C'est cette sonde du simplexe qui sera analogue aux sections POLL des algorithmes présentés antérieurement.

Suivant la sonde du simplexe, si la celle-ci est fructueuse et que la norme de l'approximation du gradient simplexe n'est pas plus petite que τh^k, l'algorithme effectue une recherche linéaire dans la direction inverse du gradient simplexe $d = -\nabla_s f(x^k)$ avec un nombre de de pas maximal spécifié par l'utilisateur. Alternativement, la direction peut-être calculée avec la résolution de $Hd = \nabla_s f(x^k)f(x^k)$, où $H \in \mathbb{R}^{n\times n}$ est une matrice Hessienne issue d'un modèle mis à jour selon une méthode de Quasi-Newton (par exemple Broyden-Fletcher-Goldfarb-Shanno (BFGS) [14, 25]),$\nabla_s f(x^k) \in \mathbb{R}^n$ le gradient simplexe et $f(x^k)$ la fonction objectif évaluée à l'itéré courant. La recherche linéaire peut s'écrire de la façon suivante qu'on

abrègera BLS pour *Backtracking Line Search*.

$$\min_m \{m \ : \ f(x^k - \beta^m d) < f(x^k), m \in \{0, \text{maxitarm}\} \cup \mathbb{N}\}. \tag{2.1}$$

où $\beta^m \leq 1$ agit comme un facteur diminuant pour déterminer la longueur du pas nécessaire, $d \in \mathbb{R}^n$ la direction de descente, m un entier servant de puissance à β et maxitarm est le nombre maximal de pas de recherche linéaire imposé par l'utilisateur. L'idée ici est de trouver un m minimal pour lequel la fonction évaluée en $f(x^k + \beta^m) < f(x^k)$.
Dans le cas ou la sonde échoue, la longueur du pas est diminuée de moitié et aucune autre recherche n'a lieu. L'algorithme est ainsi recommencé jusqu'à ce que la longueur du pas soit diminuée en deçà d'un seuil paramétré. La description de l'algorithme simplifié sort du cadre de travail précédemment imposé [13], quoique l'algorithme pourrait être écrit de façon à avoir un POLL suivi d'une SEARCH conditionnelle au succès du POLL.

Algorithme 5 Filtrage implicite (IMFIL)

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x^0 le point de départ

0. Initialisation des paramètres :

$h^0 \in (0, \infty)$	la longueur du pas initial
$V = \{\pm e_1, \pm e_2, \ldots, \pm e_k\}$	une matrice de motif
budget	le nombre d'évaluation maximal
maxitarm	nombre maximale d'itération de recherche linéaire
ϵ_{stop}	le critère d'arrêt sur le pas
$k \leftarrow 0$	le compteur d'itérations de l'algorithme
$\tau \leftarrow 0$	la tolérance admise sur le gradient

1. Boucle principale :

while $f_{\text{count}} <$ budget & $h^k < \epsilon_{\text{stop}}$ **do**
  Effectuer la sonde du gradient
  for $i = 1, \ldots, k$ **do**
    $F_i = f(x^k + h^k v_i), v \in V$
  end for
  Mise à jour de H si nécessaire
  évaluer $-\nabla_s f(x^k)$ (ou $d = H^{-1}\nabla_s f(x^k) f(x^k)$)
  $x_{\min} \leftarrow \text{argmin}(F)$
  if $\left\|\nabla_s f(x^k)\right\| \geq \tau h^k$ & $f(x_{\min}) < f(x^k)$ **then**
    $d \leftarrow -\nabla_s f(x^k)$
    if BLS est réalisable **then**
      $x^{k+1} \leftarrow x^k + \beta^m d$
    else
      $x^{k+1} \leftarrow x_{\min}$
    end if
  else
    $h^{k+1} \leftarrow h^k/2$
  end if
end while

2.6 Gestion des contraintes en DFO

Le problème (1) demande que x soit compris dans l'ensemble Ω. On peut représenter Ω de la façon suivante afin de représenter le cas ou les seules contraintes existantes sont des bornes

notées X.

$$\Omega = X = \{x \ \in \ \mathbb{R}^n \ : \ l_i \leq (x)_i \leq u_i\}$$

Où $l \in \mathbb{R}^* \cup \{-\infty\}$ représente le vecteur des bornes inférieures et $u \in \mathbb{R}^* \cup \{+\infty\}$ le vecteur des bornes supérieures. Sous cette même forme, on peut définir un problème comme étant non borné, si toutes les valeurs de l et de u prennent la valeur infinie, ce qui équivaut à $X = \mathbb{R}^n$

Si la variable x, en plus d'être limitée par des bornes, est contrainte par un ensemble d'inégalité, tel que :

$$\begin{aligned} \min_{x \in X \subset \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) \leq 0 \end{aligned}$$

On peut représenter son ensemble Ω réalisable tel que

$$\Omega = \{x \ \in \ X \ \subset \mathbb{R}^n \ : \ c(x) \leq 0\}$$

où $c : x \to \mathbb{R}^n$. Une approche pour traiter ce problème se nomme **la barrière extrême** [5]. On définit une fonction de barrière suivante

$$f_\Omega = \begin{cases} f(x) & \text{si } x \in \Omega \\ \infty & \text{sinon} \end{cases}$$

On peut alors redéfinir la fonction à optimiser comme étant f_Ω en ne se souciant plus des contraintes. Cependant, dépendemment de la forme de la boîte noire, il est possible qu'une solution $f(x)$ existe, et ce même si l'évaluation de $c(x) < 0$. On peut alors quantifier la violation de contrainte ainsi [8] :

$$h(x) = \begin{cases} \sum_{j \in J} (\max(c_j(x), 0))^2 & \text{si } x \in X \\ \infty & \text{sinon} \end{cases}$$

Avec J l'ensemble des indices des contraintes et X l'ensemble des points. Ces contraintes seraient alors relaxables. Audet et Dennis [6] proposent une façon de gérer la relaxation avec la **barrière progressive**. Afin de déterminer une hiérarchie des points selon leur valeur de $h(x)$ et $f(x)$, on définie des notions de dominance entre deux points. On dit que le point x

réalisable domine le point y réalisable si l'expression suivante est respectée

$$f(x) < f(y) \implies x \prec_f y \text{ si } x, y \in \Omega.$$

Cependant, si x est non-réalisable, il domine le point y non-réalisable si

$$f(x) \leq f(y), h(x) \leq h(y) \implies x \prec_h y \text{ si } x, y \in \Omega/X$$

avec au moins une inégalité stricte.
L'algorithme de la barrière progressive admet deux solutions courantes, soient x^{feas} la solution réalisable dont la valeur de la fonction objectif est minimale, et x^{inf}, la solution irréalisable non-dominée dont la valeur de la fonction objectif est minimale et où $h(x^{feas})$ est inférieur à h^k_{max}, un paramètre limite pour h spécifique à l'itération k.

Dans la Figure 2.2, on observe peut voir quelques points candidats. Les points pleins sont les points non-dominés et les points vides sont dominés ou exclus par la barrière h_{max}. Ceux présent dans la zone grise sont dominés par des points irréalisables pour lesquels $f(x)$ ou $h(x)$ sont inférieurs, ou encore leur valeur de leur fonction de violation $h(x)$ est supérieur à h_{max}. L'interaction entre l'utilisation des concepts de barrière et les différents algorithmes d'optimisation sans dérivées sera vue lors de la description des algorithmes.

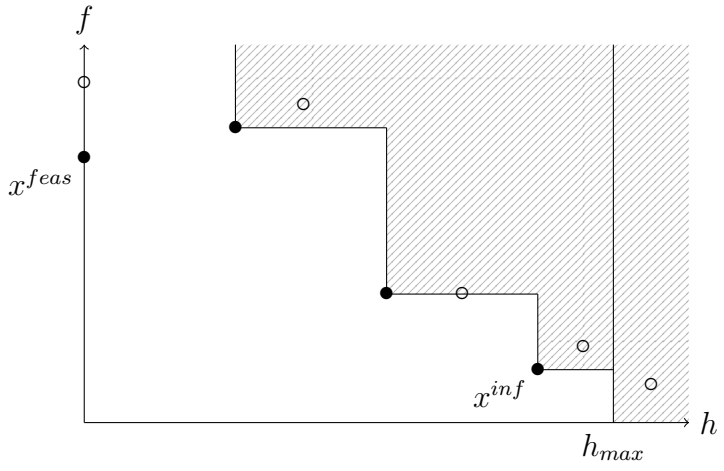

Figure 2.2 Barrière progressive, dominance et solutions courantes multiples

CHAPITRE 3 OPPORTUNISME ET ORDONNANCEMENT

Dans le contexte d'optimisation de boîte noire, chaque évaluation de la fonction objectif requiert un temps d'attente non-négligeable par l'utilisation de ressources informatiques ; il est indispensable de veiller à ce que l'algorithme choisi soit conçu avec des spécificités réduisant le nombre d'appels à la boîte noire. Dans cette optique, les étapes de POLL des algorithmes présentés précédemment sont revues.

3.1 Stratégie Opportuniste

Pour les algorithmes présentés à la section précédente, les analyses de convergence reposent sur une suite d'échecs à l'étape de POLL, tandis que les étapes de SEARCH sont d'avantage accessoires et simplement destinées à accélérer la convergence en pratique. Afin d'adapter les méthodes de recherche directe aux problèmes pouvant être traités comme des boîtes noires, il est primordial d'étudier chaque aspect pouvant entraîner une réduction d'évaluation de la fonction objectif dans la POLL sans interférer avec ses propriétés. Ainsi, l'outil fourni est soutenu par une base théorique forte prouvant l'obtention d'une solution optimale [47, 18, 5, 33, 34] et est muni d'un processus soigneusement conçu pour éviter les évaluations couteuses. Les quelques définitions suivantes seront nécessaires à l'introduction de la stratégie principale qui reste à être formellement introduite.

Définition 3.1.1 (Ensembles de recherche et de sonde)**.** *L'ensemble de recherche, désigné par S^k, est l'ensemble de points candidats pour l'étape de SEARCH. L'ensemble de sonde, désigné P^k, est l'ensemble des points candidats pour l'étape de POLL à l'itération k d'un algorithme de recherche directe.*

Par exemple, pour CS, $S^k := \{\}$ et $P^k := \{x^k + \delta^k e_i \text{ avec } i \in \{1, 2, \ldots, n \cup \mathbb{N}\}\}$.

Définition 3.1.2 (Diminution simple)**.** *Pour un algorithme de recherche directe, on dit qu'un candidat de S^k entraîne une diminution simple si*

$$\exists\ \tau\ \in S^k \text{ tel que } f(\tau) < f(x^k). \tag{3.1}$$

Analoguement, on dit qu'un candidat dans P^k entraîne une diminution suffisante si

$$\exists\ \tau\ \in P^k \text{ tel que } f(\tau) < f(x^k). \tag{3.2}$$

La présence d'un candidat dans P^k entraînant une diminution simple peut être un critère de succès pour une étape de POLL. Ce n'est pas toujours le cas, tel que présenté précédemment dans GSS, où le critère est de succès d'une POLL requiert une diminution suffisante $\exists\ \tau\ \in P^k$ tel que $f(\tau) < f(x^k) + \rho$, où $\rho \geq 0$ est la différence minimale désirée entre les deux valeurs de la fonction objectif. Outre les diminutions simple et suffisante, on introduit un autre critère caractérisant un candidat.

Définition 3.1.3 (Meilleure diminution)**.** *Pour une étape de POLL d'un algorithme de recherche directe, on dit qu'un candidat τ de P^k entraîne la meilleure diminution de la fonction objective si*

$$f(\tau) \leq f(p^k)\ \forall\ p^k \in P^k$$

Le cas traité ici est un l'obtention d'un optimum en utilisant un algorithme dans sa forme séquentielle. En pratique, il s'agit d'évaluer les points de façon séquentielle ou parallèle. Cependant, dans le cas d'optimisation de boîte noire, il est nécessaire de choisir si l'utilisation des ressources parallèles sont destinées à l'évaluation de la boîte noire, ou si elles sont dédiées à une version parallèle de l'algorithme [31, 7]. Il est intéressant de paralléliser un algorithme pour un ensemble de tâches indépendantes, par exemple dans l'optimisation de paramètres algorithmiques [3]. Dans le cas où la boite noire est lourde en calculs et nécessite de grandes ressources en parallèle pour une évaluation unique, il est envisageable d'utiliser un algorithme dans sa version en séquentielle. L'utilisation d'un algorithme de recherche directe séquentiel signifie qu'au moment initial d'une étape de POLL, $2n$ évaluations en série de la boîte noire sont possiblement à venir. Dans les algorithmes présentés, pour considérer la POLL courante comme un succès, on n'exige pas la détermination d'un candidat entrainant la meilleure diminution au sens de la définition 3.1.3 ; elle exige la détermination d'un candidat apportant une diminution simple. Ainsi, pour le premier candidat satisfaisant Léquation (3.2), l'algorithme pourrait passer à l'étape suivante en considérant la présente comme un succès. Alternativement, on pourrait aussi continuer à évaluer la fonction objectif aux autres points de l'ensemble de sonde, de façon à identifier le candidat entraînant la meilleure diminution. Ces décisions algorithmiques sont le cœur de l'étude présente.

Définition 3.1.4 (Stratégie opportuniste (ou opportunisme))**.** *La stratégie opportuniste désigne l'arrêt prématuré de l'étape SEARCH ou POLL courante d'un algorithme de recherche directe dès la première obtention d'un point τ satisfaisant le critère de succès de l'algorithme, soit la diminution simple ou la diminution suffisante.*

En opposition à la stratégie opportuniste, on définit l'idée consistant à évaluer tous les points de l'ensemble P^k ou S^k.

Définition 3.1.5 (Sonde complète et recherche complète)**.** *La sonde complète désigne l'évaluation de la fonction objectif à tous les points candidats de P^k de l'étape de* POLL *sans égard à l'identification encourue d'un candidat entrainant une diminution simple ou suffisante.*

La recherche complète désigne l'évaluation de la fonction objectif à tous les points candidats de S^k de l'étape de SEARCH *sans égard à l'identification encourue d'un candidat entrainant une diminution simple ou suffisante.*

Ces définitions concordent avec les mentions de l'opportunisme dans la littérature. Coope et Price [18] sont les premiers à utiliser le terme d'opportunisme. De prime abord, ils identifient deux cadres de travail pour des algorithmes d'optimisation sans-contraintes dont les points sont limités à des maillages, soit les cadres A et B. Les deux cadres peuvent être résumés ainsi.

Algorithme 6 Cadre algorithmique opportuniste(A) de Coope et Price

$f : \mathbb{R}^n \to \mathbb{R}$ la fonction objectif et x^0 le point de départ.

1. Déterminer un pas δ^k et une base positive ordonnée $D^k := d_1^k, d_2^k, \ldots, d_l^k$. Fixer $i \leftarrow 1$.
2. Déterminer une direction de descente $d_i^k \in D^k$ qui emmène une diminution simple de la fonction. Si une telle direction existe, mettre à jour la solution courante x^k. Sinon, essayer avec la prochaine direction dans la base positive ordonnée en incrémentant $i \leftarrow i + 1$. Si aucun i ne satisfait la condition, passer à l'étape 3.
3. Déterminer un ensemble de points finis avec une procédure arbitraire (SEARCH). Si cette procédure produit une diminution simple de la fonction objective, mettre à jour la solution. Retour à 1.

Algorithme 7 Cadre algorithmique non-opportuniste (B) de Coope et Price

$f : \mathbb{R}^n \to \mathbb{R}$ la fonction objectif et x^0 Le point de départ

1. Déterminer un pas δ^k et une base positive ordonnée D^k. $i \leftarrow 1$.
2. Déterminer la direction menant à la meilleure descente d_i^k dans D^k. Faire une recherche linéaire car cette direction mène à un succès. Mettre x^k à jour et recommencer. Si aucune direction de descente est trouvée, un minimum sur le maillage est atteint.
3. Déterminer un ensemble de points finis avec une procédure arbitraire (SEARCH). Si cette procédure produit une diminution simple, mettre à jour la solution. Retour à 1.

Les auteurs stipulent que le cadre algorithmique B est une adaptation du A et enchaînent avec la démonstration que la procédure répétée de A mène ultimement à un la détermination d'un point stationnaire de la fonction f. Ils amènent le point que l'algorithme B est beaucoup plus restreint que l'algorithme A. Par contre, c'est la routine primitive de A qui assure la convergence, malgré que le candidat entraînant la meilleure diminution n'est pas obligatoirement identifié. C'est ici qu'ils mentionnent l'opportunisme associé à ce concept algorithmique, la première instance dans littérature au meilleur de notre connaissance.

> «Pour le cadre algorithmique A, la convergence peut être démontrée seulement pour une sous-séquences de minimum locaux du maillage. Ceci est dû au fait que la recherche effectuée à l'étape 2 est opportuniste ; le premier membre rencontré dans D qui donne une descente (diminution simple) mène à un nouvel itéré.»

La concept de l'opportunisme, souvent désigné autrement dans la littérature, impacte l'analyse de convergence. Dans [47], Torczon introduit les mouvements exploratoires, sur lesquels l'analyse de convergence de GPS, et par extension CS, reposent. Deux résultats sont identifiés. En premier lieu, on statue que la convergence globale est assurée pour une méthode de GPS sur une fonction objectif f continue différentiable et sur un ensemble avoisinant un ensemble compacte. La conclusion étant que sous ces conditions, une série infinie d'évaluations mène à un point ou la norme du gradient est nulle.

$$\liminf_{x\to\infty} \|\nabla f(x_k)\| = 0 \tag{3.3}$$

Ce résultat peut être renforcé par des hypothèses plus strictes sur la norme des colonnes de la matrice génératrice Z, sur le paramètre d'ajustement du maillage τ et, plus important encore, les hypothèses de mouvements exploratoires. Initiallement, ces hypothèses se limitaient à

- La direction d est choisie $D = GZ$ et la longueur est contrôlée par δ^k
- Si un des points existants dans P^k entraîne une diminution simple, alors un mouvement exploratoire produira un pas qui entrainera une diminution simple.

Le résultat de convergence s'atteint en conservant la première hypothèse et en remplaçant la deuxième par la suivante :

- Si un des points existants dans P^k entraîne une diminution simple, alors un mouvement exploratoire produira un pas qui entrainera la meilleure descente.

Le résultat de convergence renforcé stipule :

$$\lim_{x\to\infty} \|\nabla f(x_k)\| = 0. \tag{3.4}$$

L'hypothèse sur les mouvements exploratoires est uniquement applicable si tous les points de P^k sont évalués. Il en découle que la convergence est influencée par l'opportunisme. L'utilisa-

tion de la stratégie opportuniste contredit l'hypothèse nécessaire pour la convergence forte, ainsi une implémentation de GPS opportuniste assure seulement une convergence dans le sens de la limite inférieure. Toujours dans [47], l'auteur adapte CS au cadre énoncé pour une méthode de GPS, entraînant ainsi le résultat de convergence applicable à être applicable à CS. Une situation semblable est répétée dans [34] pour GSS, soit que la convergence plus forte de l'équation (**??**) est atteignable avec la prémisse que x^{k+1} soit le candidat amenant la meilleure descente seulement. Les algorithmes présentés dans dans la section 2 ne vont pas dans les détails de leurs implémentations et la présence de l'opportunisme y est laissée floue. Conn, Scheinberg et Vincente [17] évoque explicitement la présence de l'opportunisme dans les cadres algorithmiques de méthodes de recherche directe directionnelles qu'ils utilisent.

3.2 Ordonnancement

L'utilisation de l'opportunisme entraîne que l'ordre des points dans P^k prend soudainement un rôle important. Dans l'optique de réduire le nombre d'appels à la boîte noire, on s'intéresse ici à la possibilité de réordonner P^k de façon à évaluer les candidats possiblement intéressants en premier lieu. Dans cet ordre d'esprit, Custodio et Vincente [21] tentent de déterminer des indicateurs de descente, c'est à dire une direction avec un potentiel de descente intéressant identifié à l'aide des informations obtenus à présent sur le problème.

Définition 3.2.1 (Ordonnancement)**.** *L'ordonnancement réfère à la permutation des directions, soient des colonnes de P^k ou S^k, suivant une stratégie donnée. Une stratégie guidant un ordonnancement est désignée comme stratégie d'ordonnancement.*

Dans leur étude, Custodio et Vincente identifient les dérivées du simplexe, calculables à l'aide des évaluations précédentes, compte tenu de certaines contraintes sur la géométrie de l'ensemble de points choisis [16]. Ils utilisent la distance angulaire minimale entre $-\nabla_s f(x)$ et d_i comme stratégie d'ordonnancement. Les résultats numériques montrent que, sur une version opportuniste de CS, cet ordre réfléchi des points dans P^k est grandement bénéfique si comparé à un ordre restant inchangé, surtout si combiné à d'autre stratagèmes identifiés par les auteurs. De ces stratégies on nomme l'utilisation des informations obtenues sur chaque point (en opposition à utiliser seulement les points garantissant une diminution simple) pour assurer la répartition de l'ensemble de points ainsi que l'expansion de la longueur du pas limitée à une série de succès consécutifs issues de la même direction de succès, stratégie proposée dans [31]. Custodio, Dennis et Vincente [19] réutilisent la stratégie dans le contexte d'optimisation non-lisse et observent toujours une amélioration en comparant avec un ordre arbitraire restant inchangé au cours de l'optimisation. Abramson, Audet et Dennis [1] montrent quant

à eux qu'il est possible d'avoir un ordre pour P^k permettant d'avoir une seule évaluation par étape de POLL fructueuse en utilisant les informations sur les dérivés. Lorsque Audet et Dennis introduisent MADS [5], ils spécifient que l'ordre des directions dans leurs ensembles P^k sont aléatoirement déterminés.

Définition de précéder $\prec$

3.2.1 Ordonnancement lexicographique

3.2.2 Ordonnancement en fonction du dernier succès

: =

$$\frac{d_{i,k} \cdot d_{i,k-1}}{||d_{i,k-1}||||d_{i,k}||} > \frac{d_{j,k} \cdot d_{k-1}}{||d_{k-1}||||d_{j,k}||} \implies d_{i,k} \prec d_{j,k} \tag{3.5}$$

3.2.3 Ordonnancement en fonction du modèle

:

$$\tilde{f}(p_i) < \tilde{f}(p_j) \implies p_i \prec p_j \tag{3.6}$$

3.2.4 Ordonnancement aléatoire

Hasard.

3.2.5 Ordonnancement omniscient

Stratégie pour tester "borne supérieure"

$$\tilde{f}(p_i) < \tilde{f}(p_j) \implies p_i \prec p_j \tag{3.7}$$

3.2.6 Ordonnancement négatif-omniscient

Stratégie pour tester "borne inférieure"

$$\tilde{f}(p_i) < \tilde{f}(p_j) \implies p_i \prec p_j \tag{3.8}$$

3.3 Mise en place de l'opportunisme

En forçant l'évaluation séquentielle de la fonction sur les candidats de P^k et S^k, l'opportunisme dicte la forme à prendre pour les algorithmes. Ainsi, leurs cadres proposés dans la littérature sont raffermis et limités à des formes séquentielles. Cependant, l'application de la stratégie est limitée à certains algorithmes dont la forme permet son instauration. Certains algorithmes tels que l'algorithme de Nelder-Mead [43] et les algorithmes de régions de confiance [17] en optimisation sans-dérivées possèdent un seul candidat par itération de l'algorithme. L'ordonnancement est donc incompatible avec certains algorithmes, ce qui justifie la tâche d'identifier préalablement un ensemble d'algorithmes compatibles. Le critère de compatible principal consiste à l'existence d'une étape ou une liste de points doit être évaluée, soient les étapes de SEARCH et de POLL ainsi que dans leur équivalent pour IMFIL. La section qui suit détaillera la modification des étapes de POLL pour les algorithmes de recherche directe identifiés préalablement. L'opportunisme pour les étapes de SEARCH, quoique figurant comme paramètre pour certaines implémentations [11], est laissée de la côté pour la suite du présent travail, compte tenu de l'extrême versatilité de l'étape de SEARCH.

3.3.1 CS, GPS, GSS et MADS

À des fins de généralité, les ensembles des directions de sonde utilisés dans CS et dans MADS, soient l'ensemble des directions élémentaires et $\mathbb{D}_{\Delta}^{k}$ seront substitués par D^k.

Algorithme 8 POLL opportuniste

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x^0 le point de départ

1. POLL

   Ordonner P^k selon la stratégie voulue.

   for $t \in P^k$ **do**

   if $f(t) < f(x^k)$ **then**

   $x^{k+1} \leftarrow t$

   Mise à jour de δ^k et Δ^k selon un succès de l'étape

   Poursuivre à la Terminaison.

   end if

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

   Mise à jour de δ^k et Δ^k selon un échec de l'étape

   end for

3.3.2 IMFIL

L'utilisation de l'opportunisme dans IMFIL nécessite une adaptation partielle de la méthode. Premièrement, la notion de succès pour IMFIL est différente de celle établie pour les autres algorithmes identifiés. On dira que la sonde du gradient comporte un succès si il existe un point F_i dans $\{F\}$ pour lequel la fonction objective est simplement diminuée. Pour respecter 3.2.1 sans compromis, on devrait terminer la sonde du gradient au premier succès. Cependant, IMFIL enchaîne cette sonde peu raffinée avec une descente du gradient, qui constitue apporte substance à l'algorithme et lui confère un intérêt. L'interruption trop précoce de la sonde du gradient empêchera l'obtention de l'information nécessaires pour la détermination d'une direction de descente cohérente, puisque celles-ci sont déterminées à l'aide de différences finies obtenues avec les points de F. Par exemple, à l'obtention d'un succès à une première itération, l'algorithme effectuera une descente du gradient avec $n-1$ directions nulles. On désignera cette méthode comme IMFIL-op pour IMFIL avec opportunisme pur. On devine que cette stratégie nuira au bon fonctionnement pratique de la méthode.

Algorithme 9 Sonde du gradient IMFIL-op

Avec $f : \mathbb{R}^n \to \mathbb{R}$ la fonction objectif et x^0 le point de départ
Effectuer la sonde du gradient
for $i = 1, \ldots, k$ **do**
  $F_i = f(x^k + h^k v_i), v \in V$
  if $F_i < x^k$ **then**
    $x_{\min} \leftarrow F_i$
    Poursuivre au calcul de la direction de descente
  end if
end for

On désignera cette méthode comme IMFIL-od pour IMFIL avec opportunisme décalé.

Algorithme 10 Sonde du gradient IMFIL-od

Avec $f : \mathbb{R}^n \rightarrowtail \mathbb{R}$ la fonction objectif et x^0 le point de départ
Effectuer la sonde du gradient
for $i = 1, \ldots, n$ **do**
  $F_i = f(x^k + h^k v_i), v \in V$
end for
if $f(\arg\min(F)) < f(x_k)$ **then**
  $x_{\min} \leftarrow \arg\min(F)$
  Poursuivre au calcul de la direction de descente
end if
for $i = n + 1, \ldots, k$ **do**
  $F_i = f(x^k + h^k v_i), v \in V$
  if $F_i < x^k$ **then**
    $x_{\min} \leftarrow F_i$
    Poursuivre au calcul de la direction de descente
  end if
end for

La méthode IMFIL-od ===> erreur $O(h)$ plutôt que $O(h^2)$ dans le calcul de la dérivée.

3.4 Modèles Quadratriques pour l'ordonnancement

L'ordonnancement guidé par l'utilisation modèles quadratiques élaborés dans [15] est possible avec les algorithmes présent dans NOMAD, tels que CS, GPS et MADS. Pour obtenir un modèle, on considère la base naturelle de l'espace des polynômes de degré deux et moins.

$$\xi(x) = (\xi_0(x), \xi_1(x), ..., \xi_q(x))^T = \left(1, x_1, x_2, ..., x_n, \frac{x_1^2}{2}, \frac{x_2^2}{2}, ..., \frac{x_n^2}{2}, x_1x_2, x_1x_3, ..., x_{n-1}, x_n\right)^T$$

Cette base possède $q+1 = (n+1)(n+2)/2$ éléments. Le modèle m_f de la fonction f est tel que $\tilde{f}(x) = \alpha^T \xi(x)$, $\alpha \in \mathbb{R}^{q+1}$. Pour obtenir ce modèle, un ensemble de point $Y = \{y^0, y^1, ..., y^p\}$ ayant $p + 1$ éléments est nécessaire. On cherche alors a minimiser la différence entre les valeurs de la boîte noire évaluées aux points de Y et celles du modèle. Au long de l'exécution présente ou de celles passées enregistrées dans les caches, les algorithmes évalueront la boîte noire à différents points, parmis lesquels pouront être choisis les $p = q$ points nécessaires à

l'élaboration du modèle. Les points devront satisfaire la propriété qui valide le modèle :

$$B(\xi, Y)\alpha = f(Y)$$
$$f(y) = (f(y^0, f(y^1), ..., f(y^p))^T$$
$$B(\xi, Y) = \begin{bmatrix} \xi_0(y^0) & \xi_1(y^0) & \dots & \xi_q(y^0) \\ \xi_0(y^1) & \xi_1(y^1) & \dots & \xi_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \xi_0(y^p) & \xi_1(y^p) & \dots & \xi_q(y^p) \end{bmatrix}.$$

Ce système peut être résolu seulement si $p = q$ et si la matrice est de rang pleine. Dans le cas où $p \geq q$, on tentera de résoudre le problème de minimisation suivant :

$$\min_{\alpha \in \mathbb{R}} \quad \|B(\xi, Y)\alpha - f(Y)\|^2 \quad .$$

Dans le cas où $p < q$, on minimisera le même problême mais en régularisant le problème à l'aide d'une interpolation dans le sense de la norme de Frobenius minimale [41, 20]. La norme de Frobenius pour une matrice A est définie par :

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i,j}|}.$$

Il est possible de réarranger $\tilde{f}(x)$ de façon à l'illustrer comme une fonction quadratique en utilisant la notation précédente mais en divisant le modèle en ses expressions linéaires et quadratiques :

$$\tilde{f}(x) = \alpha_L^T \xi_L(x) + \alpha_Q^T \xi_Q(x).$$

L'indice L dénote les termes linéaires et d'ordre 0 de $\xi(x)$, au compte de $n + 1$, soient les n variables et le terme de degré 0, $\xi_0(x) = 1$. L'indice Q dénote les termes quadratiques au compte de $(n + 1)(n + 2)/2 - (n + 1) = \frac{n(n+1)}{2}$. Ainsi, on peut réecrire la quadratique en trois termes, soient le terme constant, le terme des composantes linéaires et le terme des composantes quadratiques :

$$\tilde{f}(x) = c + g^T x + \frac{1}{2} x^T H x.$$

Avec $g \in \mathbb{R}^n$ et la matrice $H \in \mathbb{R}^{n,n}$ la matrice Hessienne symétrique du modèle. Avec un problème sous déterminé où $p < q$, on choisira le modèle tel que :

$$\begin{aligned} \min_{H \in \mathbb{R}^{n,n}} \quad & \|H\|_H^2 \\ \text{sujet à} \quad & c + g^T y^i + \frac{1}{2}(y^i)^T H(y^i) = f(y^i) \qquad i = 1, \ldots, p. \end{aligned}$$

On entend ici minimiser l'influence des termes quadratiques pour diminuer l'amplitude du modèle entre les points de Y utilisés. Par exemple, pour un problème de dimension n avec $p \leq (n+1)$, on résoudera seulement la portion linéaire de $\alpha^T \xi(y^i) = f(y^i)$ pour ainsi laisser $h_{i,j} = 0, i, j = 1 \ldots n$ et donc $\alpha_Q = 0$.

CHAPITRE 4 RESULTATS NUMÉRIQUES

4.1 Outils de comparaison et problèmes

Les premieres recherches visant à comparer les performances de solveurs sur un problème unique ou une série de problème utilisaient des métriques facilement quantifiables telles que le nombre d'appels à la fonction [12] ainsi que le temps de résolution en secondes [40]. Cependant, ces indicateurs sont moins appropriés lorsqu'on est en situation de résolution d'un grand nombre de problèmes puisqu'ils sont spécifiques à un problème, ce qui rends le banc d'essai très volumineux. Ainsi, certains outils ont été dévloppés pour analyser une série de problème test avec des dimensions et des spécificités différentes.

4.1.1 Graphe et test de convergence

Pour comparer la performance d'un ensemble d'algorithme sur un seul problème, on aura recours au graphe de convergence. Le graphe de convergence trace la courbe de la meilleure valeur de la fonction objectif obtenue en fonction du nombre d'appel à la fonction donné.
En DFO, rien n'assure que le solveur arrivera à l'optimum global, ou encore qu'un optimum soit atteint dans le budget alloué. De plus, en situation de résolution de boîte noire, il est impossible de connaître l'optimum théorique du problème, et ce si celui-ci existe. On aimerait tout de même quantifier la performance d'un solveur. Il est donc entendu dans la littérature de faire appel à un test de convergence qui mesure la capacité à l'algorithme à s'approcher de la meilleure solution obtenue par l'ensemble des algorithmes à l'étude, tel que le suivant issu de [39] :

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L)$$

Avec x_0 le point initial, x le point courant, f la fonction à optimizer, $\tau \geq 0$ un seuil de tolérance et f_L la meilleure solution courante trouvée par un solveur sur ce problème. Le choix de f_L peut-être soit la meilleure solution connue, ou encore trouvée par les solveurs à l'étude. Le test de convergence peut être utilisé à plusieurs sauces, soient à savoir si un solveur atteint au moins une solution proche de la meilleure en utilisant un τ grossier, par exemple $\tau = 0.1$, ou encore à savoir si le solveur est capable d'obtenir f_L en utilisant $\tau = 0$. Il est possible d'observer la satisfaction du test de convergence sur un graphe de convergence en regardant quand la courbe d'un certain solveur atteint une zone de $\tau(f(x_0) - f_L)$ précédent l'ordonné représentant f_L. Il est possible que le test ne soit jamais réussi si on observe la résolution avec un budget d'évaluation fixe, ce qui sera nécessaire pour des boîtes noires

bruitées, pour lesquelles la convergence admise par l'algorithme ne dépends pas d'une autre résolution du problème.

4.1.2 Profil de performance

Un outil établi par la suite pour la comparaison de solveurs, pas nécessairement des solveurs d'optimization sans-dérivées, est le profil de performance [23]. La performance est donnée par une mesure $t_{p,s}$, qui peut être par exemple le nombre d'évaluations nécessaire pour l'obtention du minimal global, pour une paire de p un problème, et s un solveur. Pour la DFO, on choisira le nombre d'itérations nécessaire à la satisfaction du test de convergence énoncé précédemment. Afin de comparer les solveurs sur chaque problème, on défini le ratio de comparaison suivant

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}$$

On peut alors définir le profil de performance d'un solveur sur un ensemble de problème P ainsi

$$\rho_s(\alpha) = \frac{1}{|P|}\text{size}\{p \in P : r_{p,s} \leq \alpha\}$$

Pour un α, soit un indice de performance tel que le ratio de comparaison $r_{p,s}$, on peut savoir la proportion de problème que chaque solveur a su résoudre en se rapprochant à un facteur de $(1-\tau)$ de la meilleure solution trouvée par l'ensemble des solveurs pour chaque problème.Le test de convergence varie selon la nature des problèmes à résoudre. On peut utiliser un profil de performance pour des outils d'optimisation avec dérivés avec un $t_{p,s}$ qui agit sur les informations de premier ordre de la fonction, tel que le nombre d'évaluations pour atteindre x tel que $f'(x) = 0$. Ainsi, $\rho_s(\tau)$ revient à être la probabilité que le ratio de performance $r_{p,s}$ du solveur s soit à un facteur τ du meilleur ratio possible. Une lacune des profils de performance est que le ratio utilisé n'est pas en mesure de prendre en compte la dimension du problème.

Il est important de noter que les profils de performance ont une faille tel qu'élaboré dans [27]. Les auteurs concluent au terme d'une étude que l'utilisation de f_L dans le test de convergence implique que la simple observation des courbes sur un graphe n'est pas suffisante pour classes les algorithmes. Puisque la comparaison se fait avec le nombre d'itération minimal observé, il se peut que, lorsqu'on compare enrte-eux deux solveurs qui n'ont pas performés en tant que meilleur, l'ordre apparent ne soit pas celui qu'on observerait si on éliminait du graphe les données du meilleur. On se doit alors d'utiliser un autre outil pour mesurer les performances des solveurs sur un ensemble de données.

4.1.3 Profil de données

Moré et Wild proposent [41] une définition d'un profil de donnée,

$$d_s(\alpha) = \frac{1}{|P|}\text{size}\{p \,\epsilon\, P : \frac{t_{p,s}}{n_p + 1} \leq \alpha\}$$

avec t_p le nombre d'évaluations pour atteindre la convergence nécessité par le solveur s sur le problème p. Le facteur $n_p + 1$ est le nombre d'évaluations nécessaire pour le calcul de l'estimation finie d'un gradient pour un problème de taille n_p. Il est alors possible de mesurer la performance relative des solveurs comme une fonction du budget d'évaluations, sans la comparaison que le ratio de performance imposait. Ainsi, pour un α, soit une quantité de $n + 1$ évaluations de la fonction objectif, on peut savoir la proportion de problèmes que chaque solveur a su résoudre en se rapprochant à un facteur de $(1-\tau)$ de la meilleure solution trouvée par l'ensemble des solveurs pour chaque problème. L'avantage du profil de données sur le profil de performance est qu'on peut dorénavant quantifier la performance du meilleur algorithme puisque celui-ci n'est plus comparé à lui-même.

4.1.4 En optimisation sous-contraintes

4.2 Problèmes

4.2.1 Ensemble de problèmes Moré-Wild

L'échantillon de problèmes caractéristiques $\mathcal{P}$ utilisé est celui issu de [41]. Cet ensemble de problèmes a été réutilisé dans la communautée [15, 48], ce qui justifie son utilisation comme base de problème analytique pour la comparaison d'algorithme ou de stratégies algorithmiques.

L'ensemble de problème est déclinée de 22 fonctions non linéaires de moindres carrés, lesquelles sont tirées de la collection CUTEr [28]. Ces 22 fonctions sont remaniées pour donner un ensemble de 53 problèmes. L'indice k_p pour un problème $p \in \mathcal{P}$ fait référence à la fonction de base issue de CUTEr utilisée pour ce problème.

Chaque problème possède trois autres paramètres en plus de l'indice k_p, soient n_p pour la dimension du problème, m_p de nombre de composantes du problème et le paramètre binaire x_p, pour lequel, si activé, le point de départ x_s subit une homotéthie de facteur 10. L'ensemble $\mathcal{P}$ contient 53 problèmes avec un vecteur (k_p, n_p, m_p, s_p) unique. Aucune fonction sous-jacente n'est surreprésentée puisque au plus six problèmes possèdent le même k_p. Les bornes sur les

paramètres vont telles que

$$1 \leq x_p \leq 22, \quad 2 \leq n_p \leq 12, \quad 2 \leq m_p \leq 65, \quad s_p \in \{0, 1\}, \quad p = 1, \ldots, 53$$

avec $n_p \leq m_p$.
La structure par morceaux des problèmes de $\mathcal{P}$ permet de dériver l'ensemble en d'autres classes de problèmes. On entend ainsi permettre la comparaison d'algorithmes ou de stratégies algorithmiques sur différentes classes de problèmes. Les classes utilisées dans [41] qui sont réutilisées ici sont : les problèmes lisses, les problèmes lisses définis par partie et les problèmes bruités. Les problèmes lisses sont formés tels que :

$$f(x) = \sum_{i=1}^{m} (f_i(x))^2.$$

Les problèmes non-différentiables sont obtenus à l'aide d'une transformation apportée à l'expression des problèmes lisses telle que :

$$f(x) = \sum_{i=1}^{m} |f_i(x)|.$$

Ainsi, l'expression du gradient $\nabla f_i(x)$ est inexistante lorsque $f_i(x) = 0$. Afin d'assurer que chaque déclinaison de problème possède un minimum global, pour l'ensemble $k_p = [8, 9, 13, 16, 17, 18]$, la fonction est redéfinie dans l'orthan positif tel que

$$f(x) = \sum_{i=1}^{m} |f_i(x_+)|.$$

Les problèmes bruités sont obtenus à l'aide d'une transformation apportée à l'expression des problèmes lisses, soit multipliant un terme induisant un bruit à la fonction , tel que :

$$f(x) = (1 + \epsilon_f \theta(x)) \sum_{k=1}^{m} f_i(x)^2.$$

Le terme $\theta(x)$ est issu d'une composition d'une fonction trigonométrique $\theta_0(x)$ avec un polynôme de Chebyshev de degré 3 tel que $T_3(\alpha) = \alpha(4\alpha^2 - 3)$

$$\theta_0(x) = 0.9 \sin(100||x||_1) \cos(100||x||_\infty) + 0.1 \cos(||x||_2)$$
$$\theta(x) = T_3(\theta_0(x)) = T_3(0.9 \sin(100||x||_1) \cos(100||x||_\infty) + 0.1 \cos(||x||_2))$$

Cette composition élimine la périodicité de θ_0. La formulation est aussi composée du terme ϵ_f, le niveau du bruit relatif, qui est fixé à $\epsilon_f = 10^{-3}$.

4.2.2 Problèmes analytiques contraints

tableau des problèmes avec sources

4.2.3 STYRENE

description + pre analyse des stratégies

4.2.4 LOCKWOOD

description + pre analyse des stratégies

4.3 Méthodologie

4.3.1 Méthodologie commune

Les valeurs de $\tau = 0.1$,$\tau = 0.01$ et $\tau = 0.001$ sont utilisées pour les différents profils lors des résolutions des problèmes de Moré-Wild. nomad bla bla pour CS GPS MADS

4.3.2 CS

CS a été obtenu a l'aide d'un tel paramètrage de NOMAD...

4.3.3 GPS

GPS a été obtenu a l'aide d'un tel paramètrage de NOMAD...

4.3.4 MADS

Dans le but d'oberserver en premier lieu les impacts des différentes stratégies sur le coeur de MADS on désactive les options suivantes, de façon à isoler la sonde

4.3.5 MADS par défaut de NOMAD

Réactivons les paramètres par défaut de nomad pour voir l'impact de l'ordonnancement sur le logiciel sans égard à l'isolement de la sonde.

4.3.6 GSS

tel tel paramètre avec GSS, opportunisme déja implanté dans hopspack + paramètre (determination du scaling volée dans dynamic scaling)

4.3.7 Implicit Filtering

toutes les gogosses que j'ai fait pour que son code roule

4.4 Comparaison des stratégies

4.4.1 CS

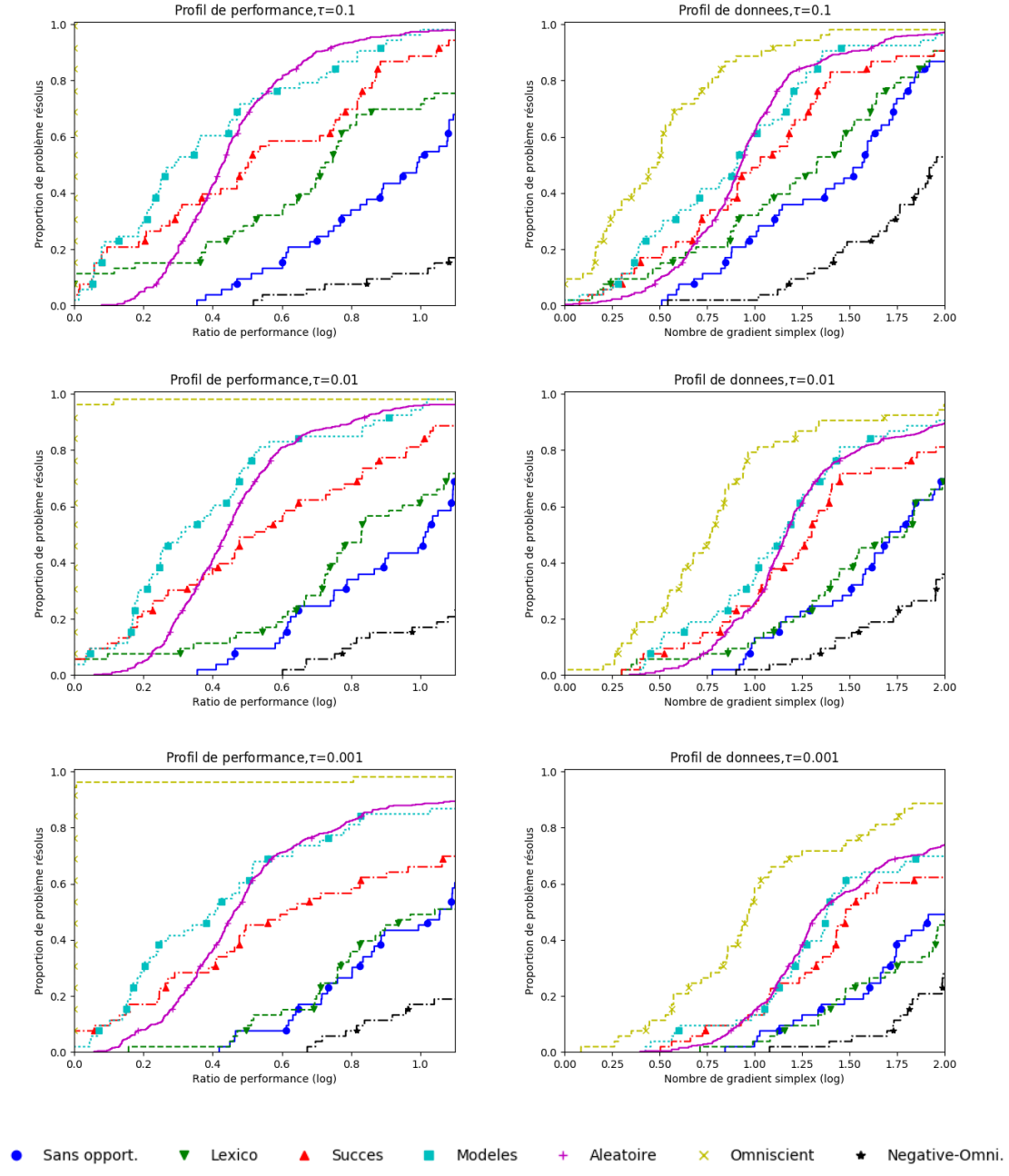

Figure 4.1 Comparaison sur problèmes lisses de Moré-Wild avec CS

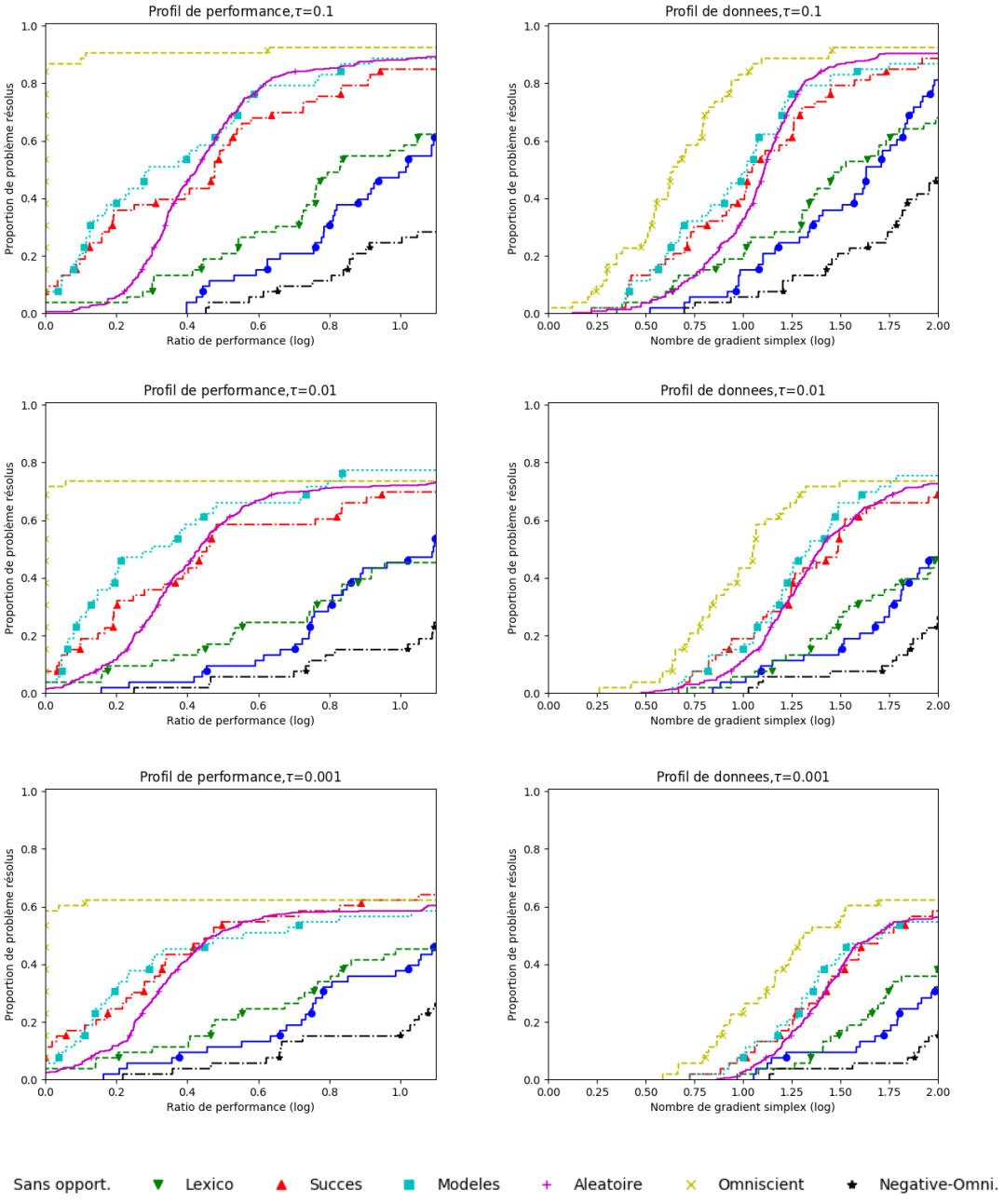

Figure 4.2 Comparaison sur problèmes non-differentiables de Moré-Wild avec CS

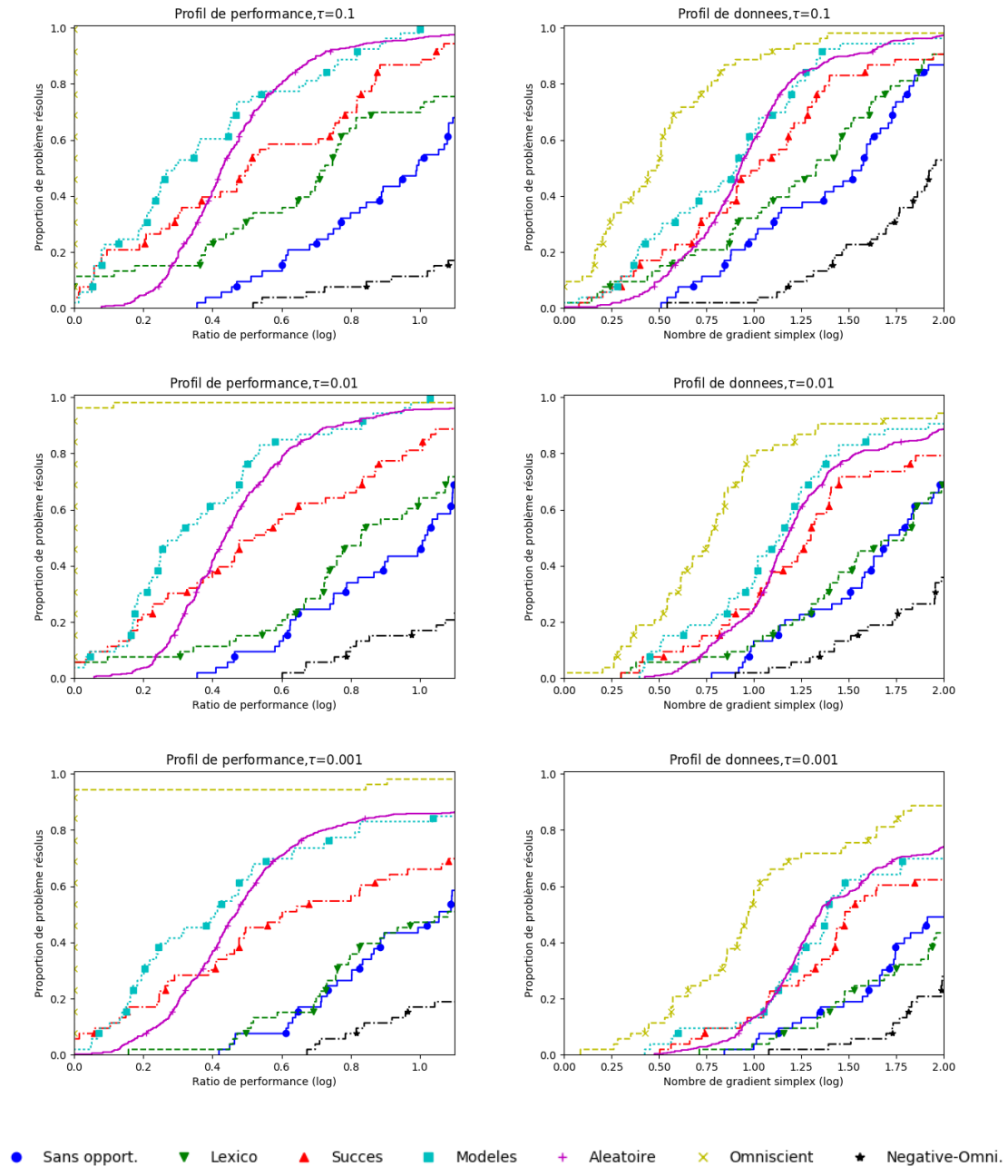

Figure 4.3 Comparaison sur problèmes bruités de Moré-Wild avec CS

La recherche par coordonnée offre une opportunité sans pareille d'observer l'impact de l'ordonnancement. Puisqu'il s'agit essentiellement d'une étape de sonde rudimentaire, elle sera très sensible à l'opportunisme et à la stratégie d'ordonnancement qui le guidera. On s'attend ici à des profils très distincts. Sur les problèmes lisses, les algorithmes se comportent en moyenne de façon plus stable. On en déduit qu'il ne sera pas pertinent de qualifier une stratégie d'ordonnancement spécialement sur son comportement en situation d'optimisation de problème lisse ; les déductions dans cette section seront plutôt de nature générale.

Outre de rappeler la pertinence des profils de données, les profils de performances montrent que la stratégie aléatoire et la stratégie d'ordonnancement par modèles sont de performances comparables. On peut y comprendre ici que les stratégies sont uniquement comparées à la stratégie omnisciente pour la tolérance la plus haute $\tau = 0.1$. La marge d'erreur est aussi limitée pour les autres valeurs de $\tau = 0.01$ et $\tau = 0.001$, alors que pour près de 95% on utilise la stratégie omnisciente pour comparer le résultat. Dans cette situation, on peut prendre directement l'allure du profil pour hierarchiser les stratégies sur l'ensemble de problème sans se soucier de l'erreur induite par la nature des profils de performance [27]. Sur le profil de performance avec $\tau = 0.1$, on voit que la stratégie d'ordonnancement en fonction du dernier succès est supérieure à la stratégie aléatoire jusqu'à un ratio de performance avoisinant 2, après coup elle est reléguée à la quatrième plus performante. Malgré que cette stratégie soit plus raffinée que la stratégie aléatoire, elle ne prendra avantage de la forme du problème que si celui-ci n'est pas composé de plusieurs minimums locaux et de points de selle. On pourrait conclure que l'ensemble de problèmes utilisé ici possède une proportion de problèmes dont la structure n'est pas idéale pour un ordonnancement basé uniquement sur le succès précédent. La stratégie lexicographique n'est pas comparable en terme de performance. La stratégie pousse l'algorithme à épuiser ses sources de directions de déscente une par une. Au fil des évaluations, les directions de déscente déjà épuisées seront évaluées en début de sonde, et celles prometteuses seront toujours à la fin de la liste, ce qui aura pour effet de décaler les succès de plus en plus au cours du déroulement de l'algorithme. Pour sa part, la sonde complète ne peut pas compétitionner en observant le profil de performance, puisqu'elle prends un nombre n fois le nombre d'évaluation nécéssité par la stratégie omnisciente. La stratégie négative omnisciente n'est pas appropriée dans la comparaison avec un profil de performance. Ces tendances sont observables aussi pour les ratios de tolérance moins élevés, qui figurent dans le deuxième et le troisième profil de performance. On en conclu que les solutions trouvées avec toutes les méthodes sont soit exacte à 0.01% ou alors à plus de 10% de différence de celle déterminée avec la stratégie omnisciente.

Dans le cas des tests avec cette famille d'algorithme, l'utilisation de la stratégie omnisciente requiert le traçage de profils de données pour bien pouvoir comparer les stratégies. Ce type de métrique est choisi plutôt que les profils de performance car la nature de leur abscisse est relative au nombre d'itérations avant d'atteindre le minimum, permettant ainsi de juger de la performance d'un algorithme indépendemment de ses concurrents. Ainsi, les profils de données permettent de mieux distinguer le meilleur performant sans qu'il soit joint aux axes, tel qu'illustré dans les trois profils de performances de la figure précédente. Nous savons d'ailleurs des profils de performance que la stratégie omnisciente donne toujours le meilleur rendement. Ainsi, on peut se fier directement à l'apparence des courbes.

Premièrement, la stratégie omnisciente semble toujours la meilleure, alors qu'on peut toutefois déterminer avec quelle ampleur elle domine les autres. On explique que son ordonnée n'atteint pas 1.0 par le fait que seulement jusqu'à $70 \times n + 1$ évaluations sont illustrées. Pour des problèmes ayant jusqu'à $n = 12$ variables tels qu'il en existe dans la collection de Moré-Wild utilisée, on a que 70 gradients simplex correspondent à 910 évaluations de la boîte noire, qui peuvent ne pas être suffisantes pour l'obtention de la meilleure solution obtenue. On observe que les comparaisons des stratégies réalistes faitent avec les profils de performance ne sont pas remis en question avec les profils de données outre la stratégie omnisciente.

4.4.2 GPS

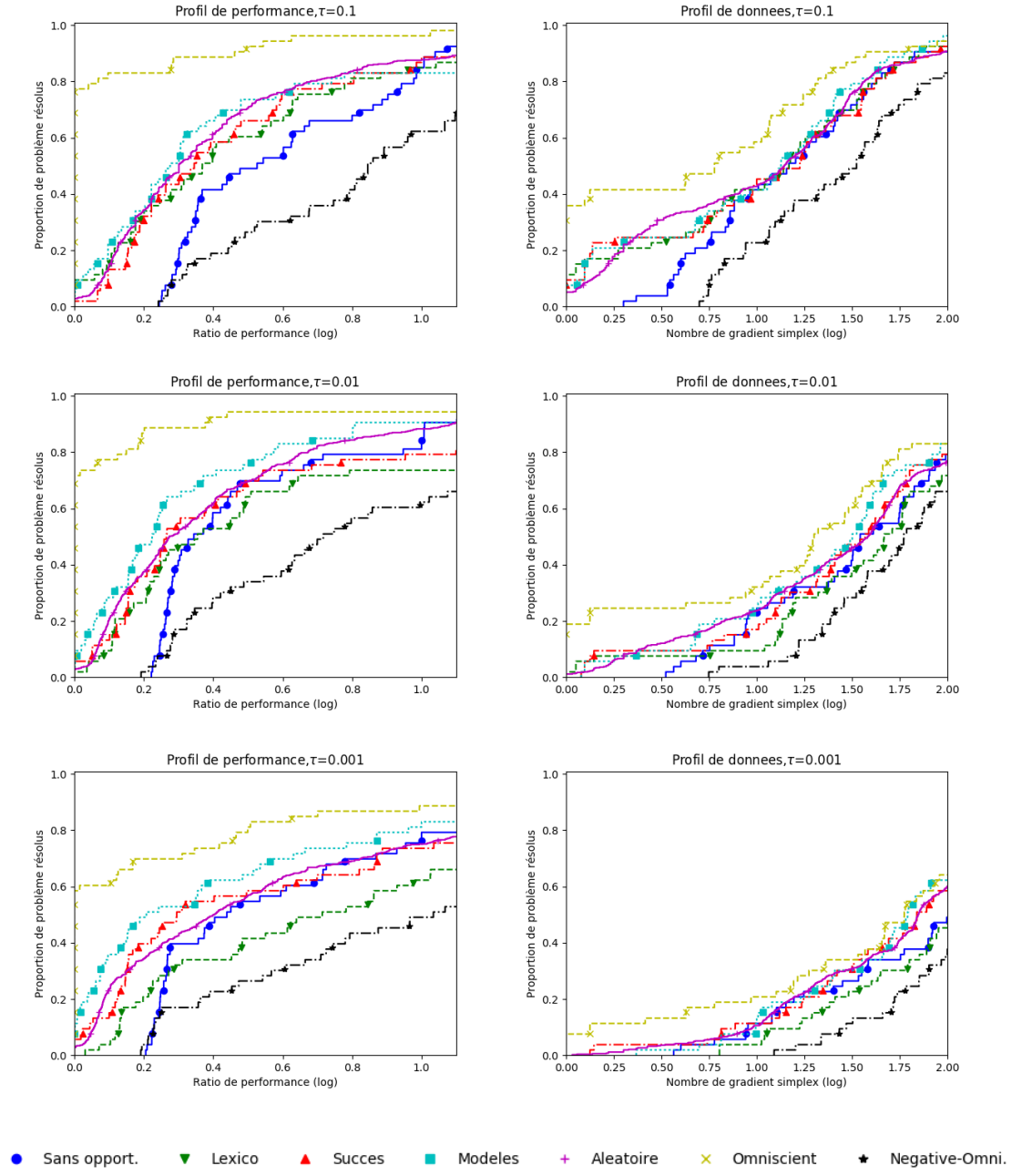

Figure 4.4 Comparaison sur problèmes lisses de Moré-Wild avec CS

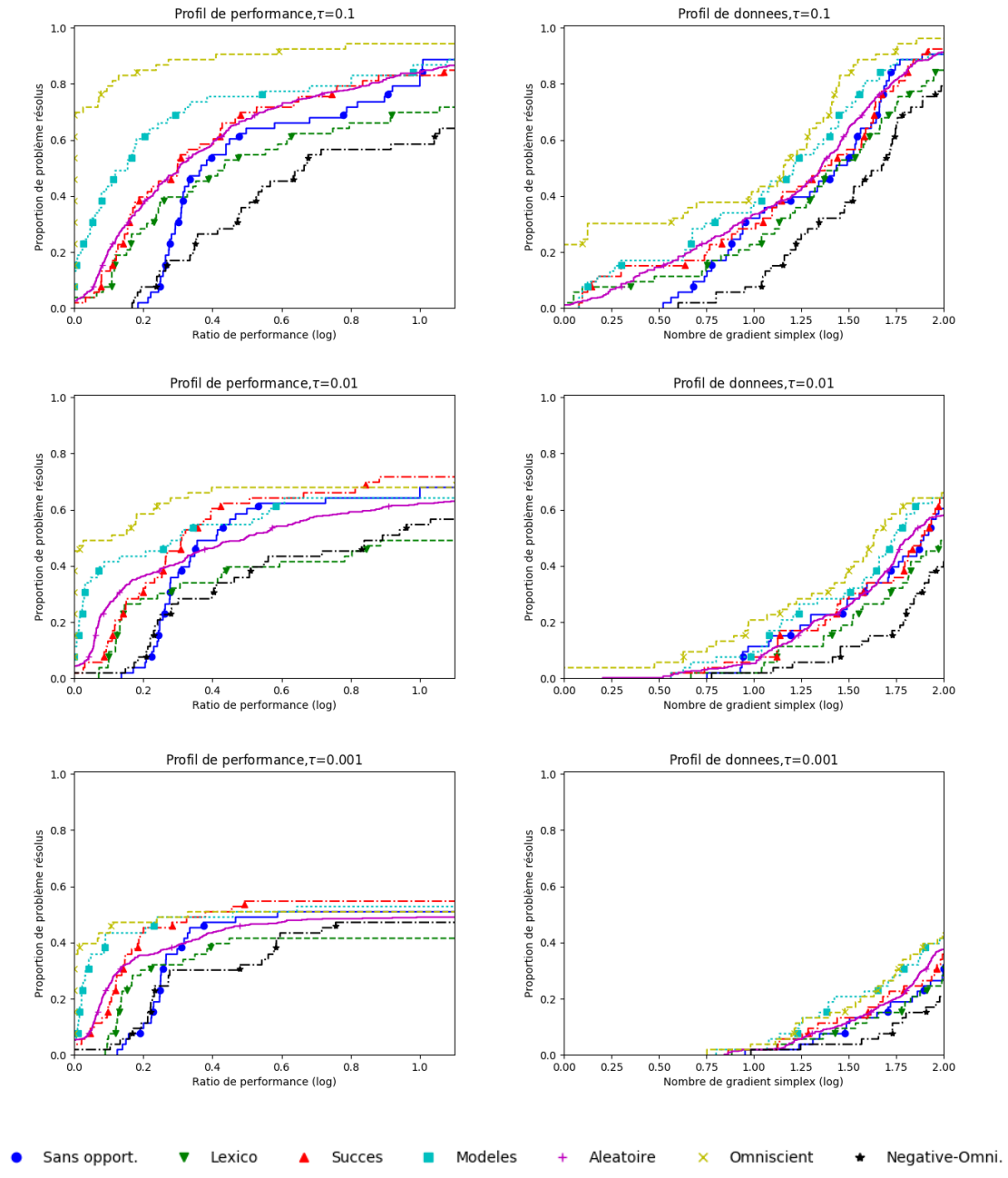

Figure 4.5 Comparaison sur problèmes non-differentiables de Moré-Wild avec CS

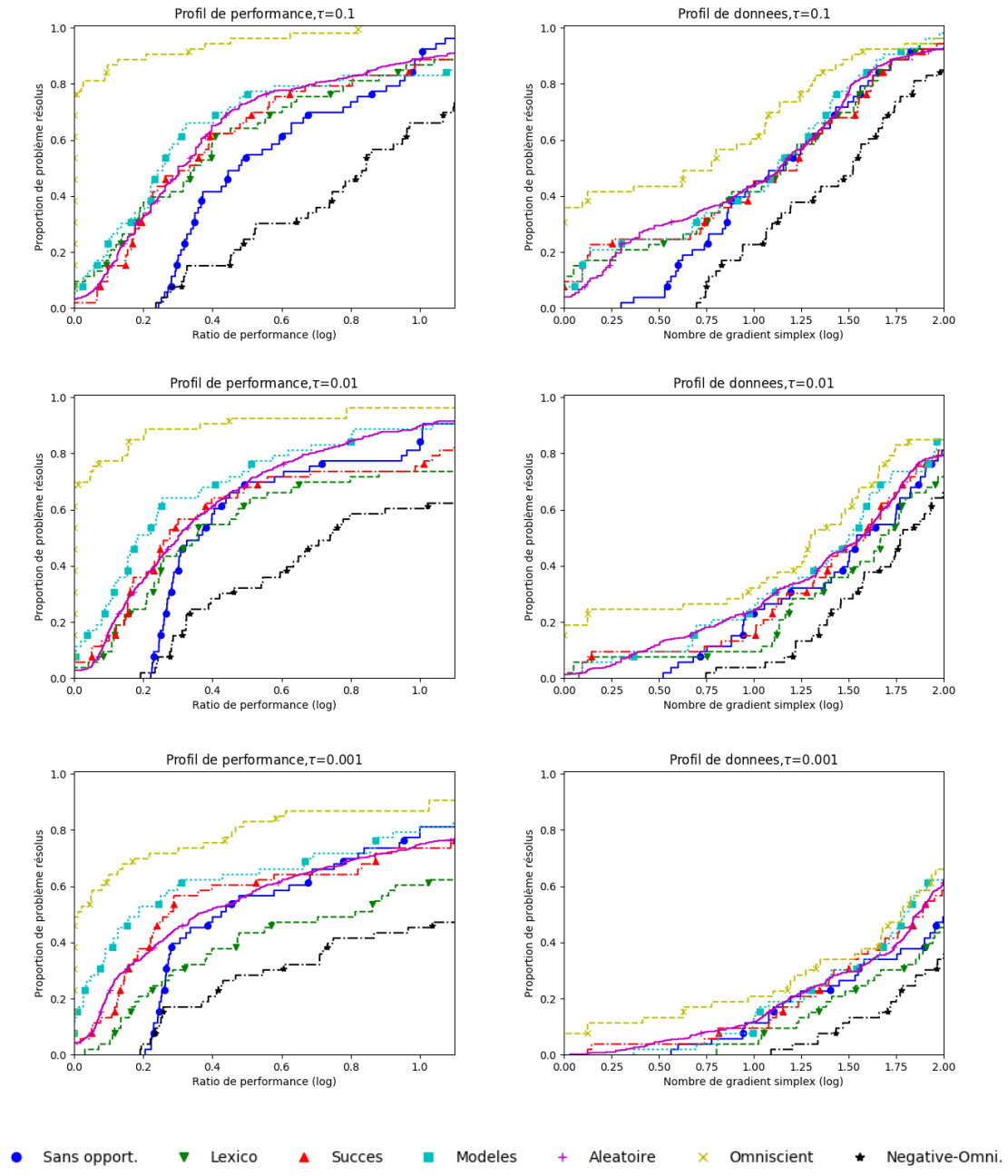

Figure 4.6 Comparaison sur problèmes bruités de Moré-Wild avec CS

Commentaires Commentaires

4.4.3 MADS

Moré-Wild

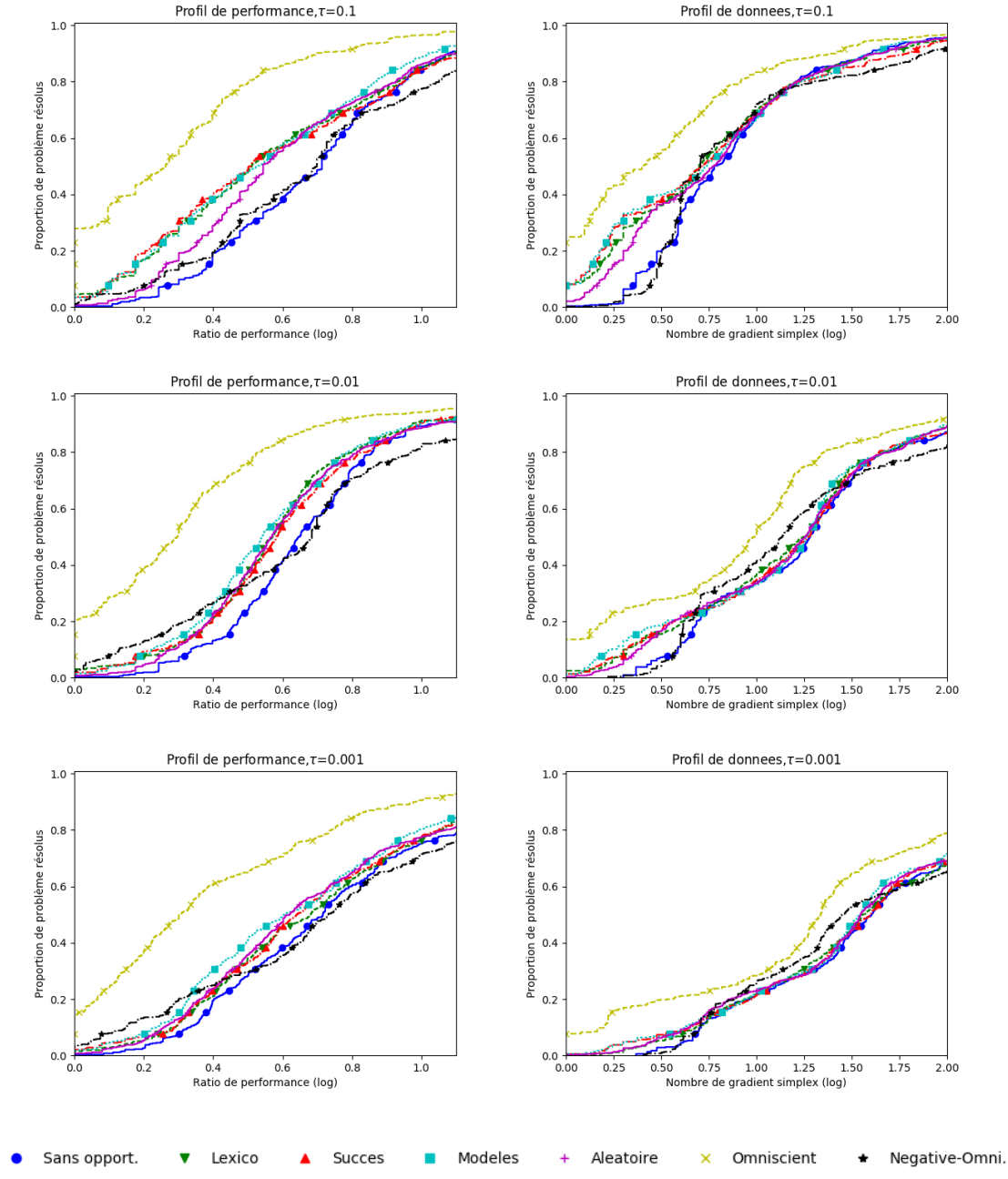

Figure 4.7 Comparaison sur problèmes lisses de Moré-Wild avec MADS

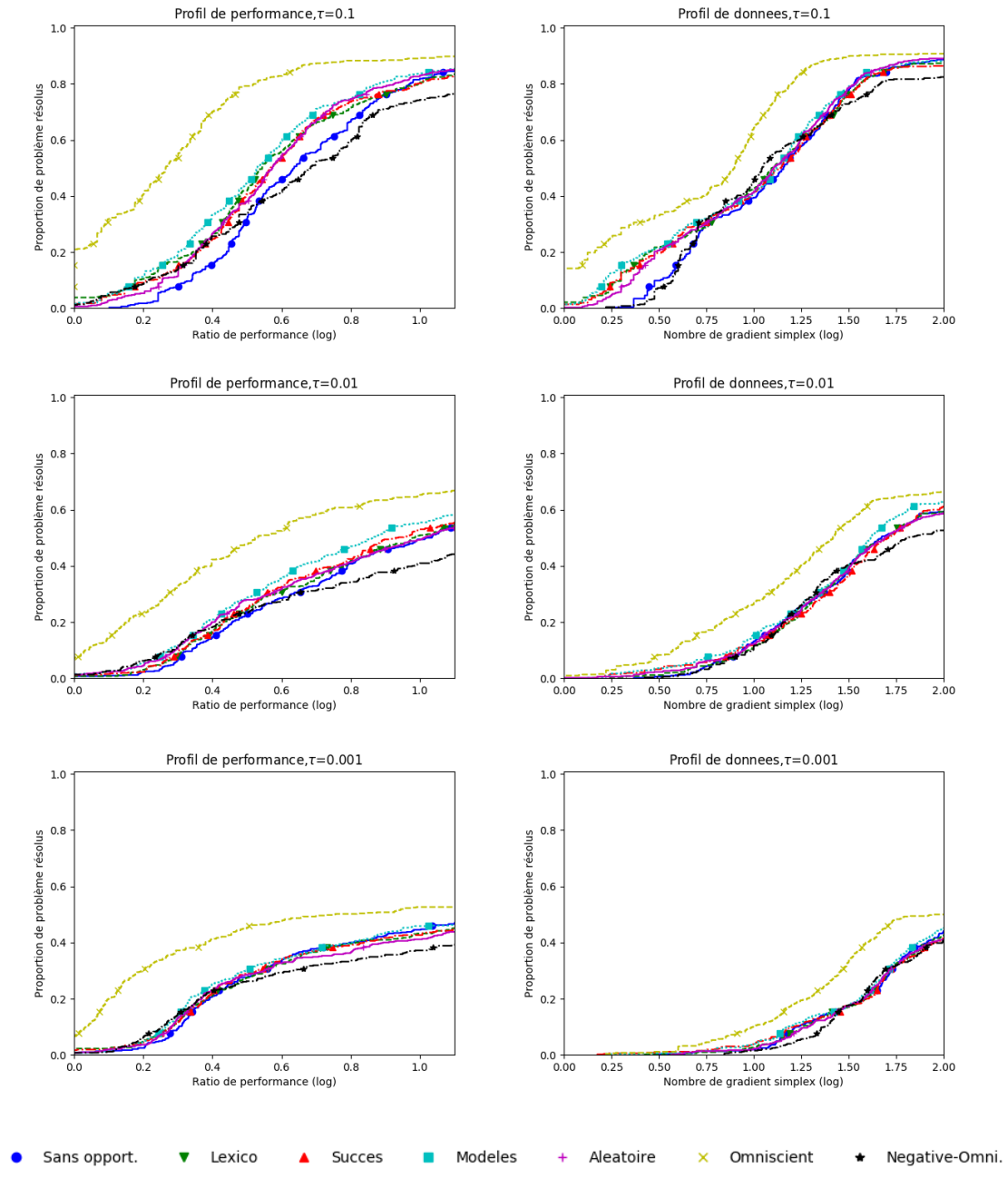

Figure 4.8 Comparaison sur problèmes non-differentiables de Moré-Wild avec CS

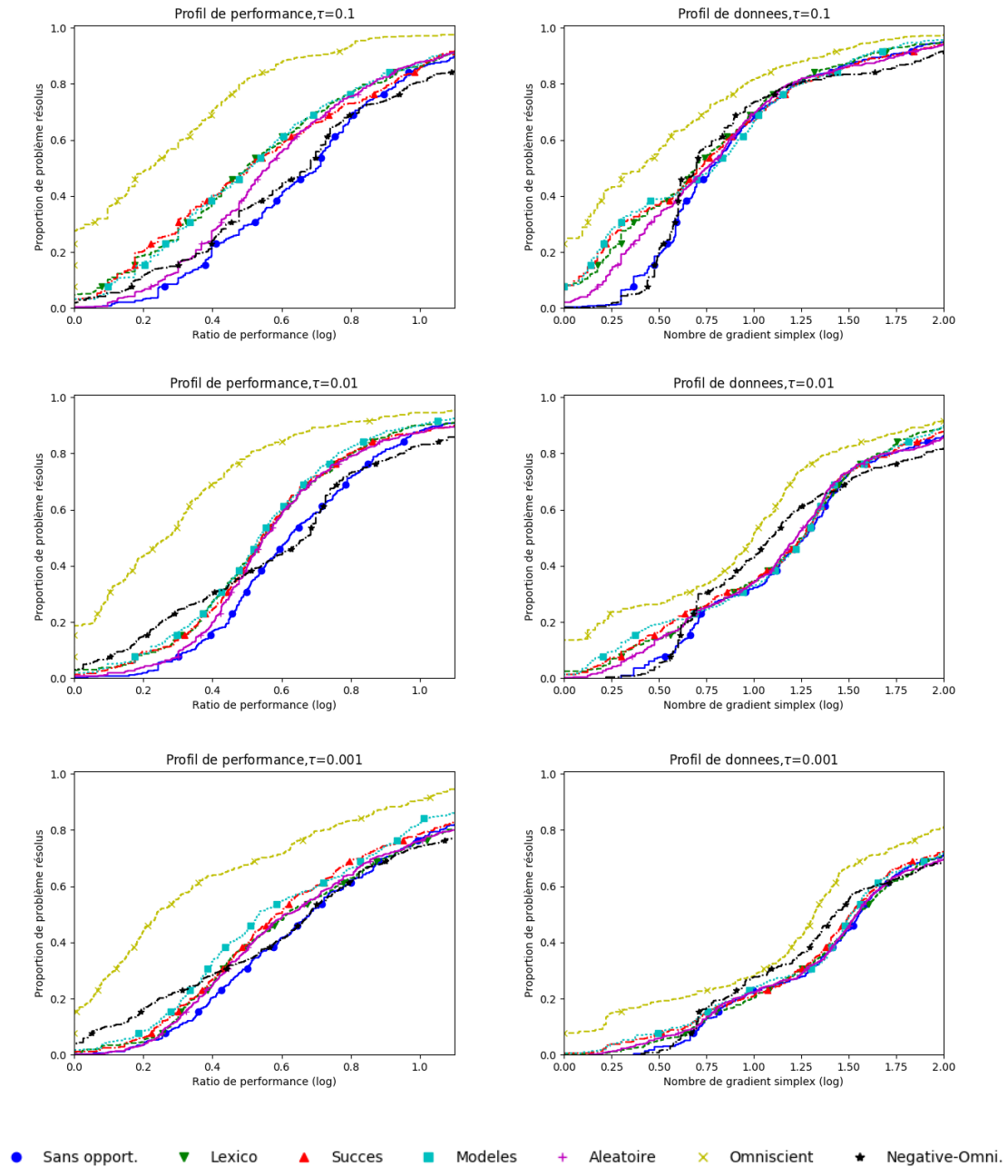

Figure 4.9 Comparaison sur problèmes bruités de Moré-Wild avec CS

Commentaires Commentaires

STYRENE

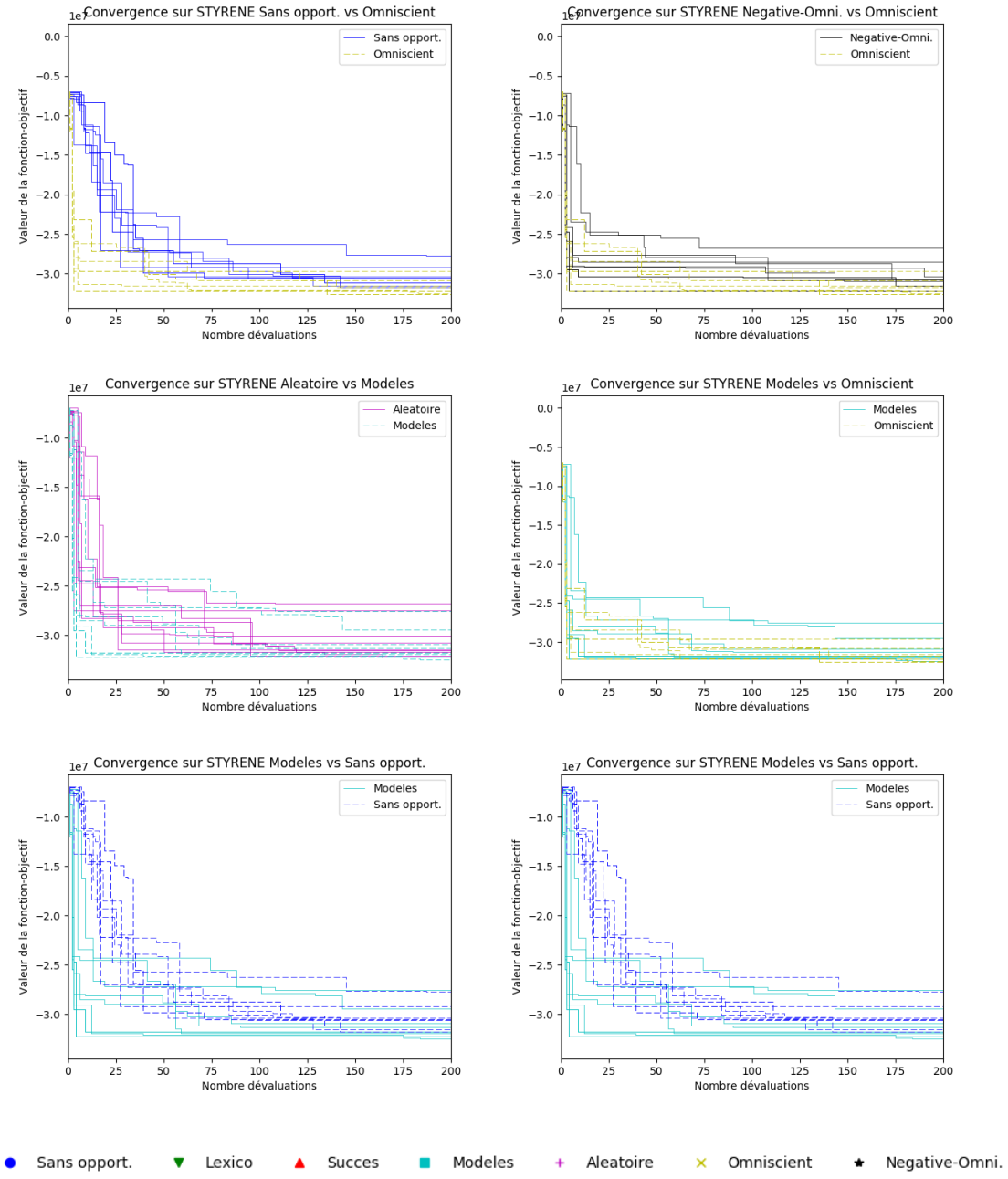

Figure 4.10 Comparaison sur STYRENE avec MADS

Commentaires CommentairesCommentaire sCommentairesCommentaire sCommentairesCommentairesCommentai resCommentairesCommentairesCommentairesCommentairesCommentairesComm entairesCommentairesCommentairesC ommentairesCommentairesCommentaires-CommentairesCommentairesCommentair esCommentairesCommentairesCommen tairesCommentairesCommentairesCommentairesCommentairesCommentairesComme ntairesCommentairesCommentaires CommentairesCommentairesCommentairesCommentairesCommentaires-CommentairesC ommentairesCommentairesComment airesCommentairesCommentairesCommentairesCommentairesCommentair esCommentaires

4.4.4 MADS de NOMAD

Moré-Wild

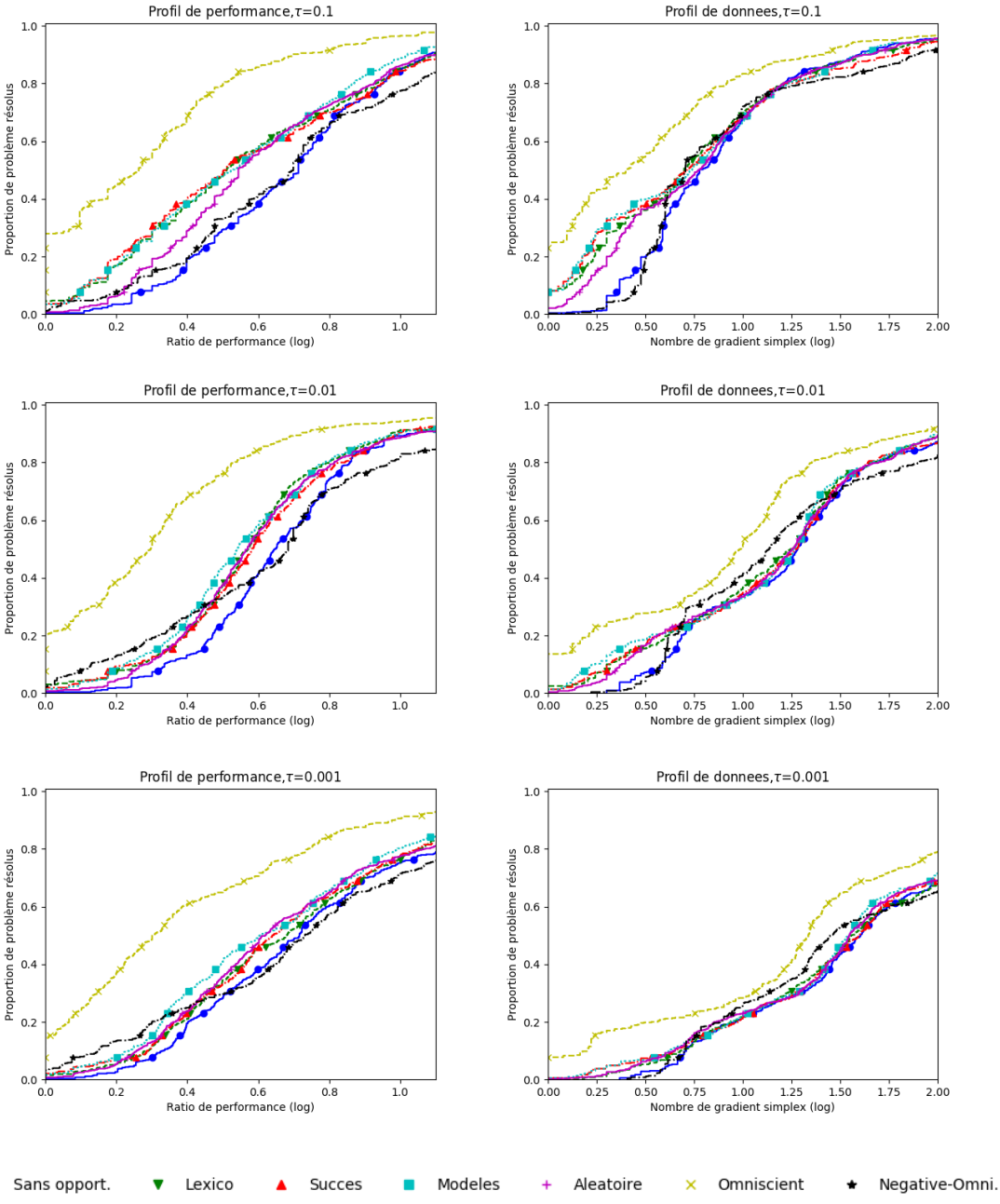

Figure 4.11 Comparaison sur problèmes lisses de Moré-Wild avec MADS par défaut de NOMAD

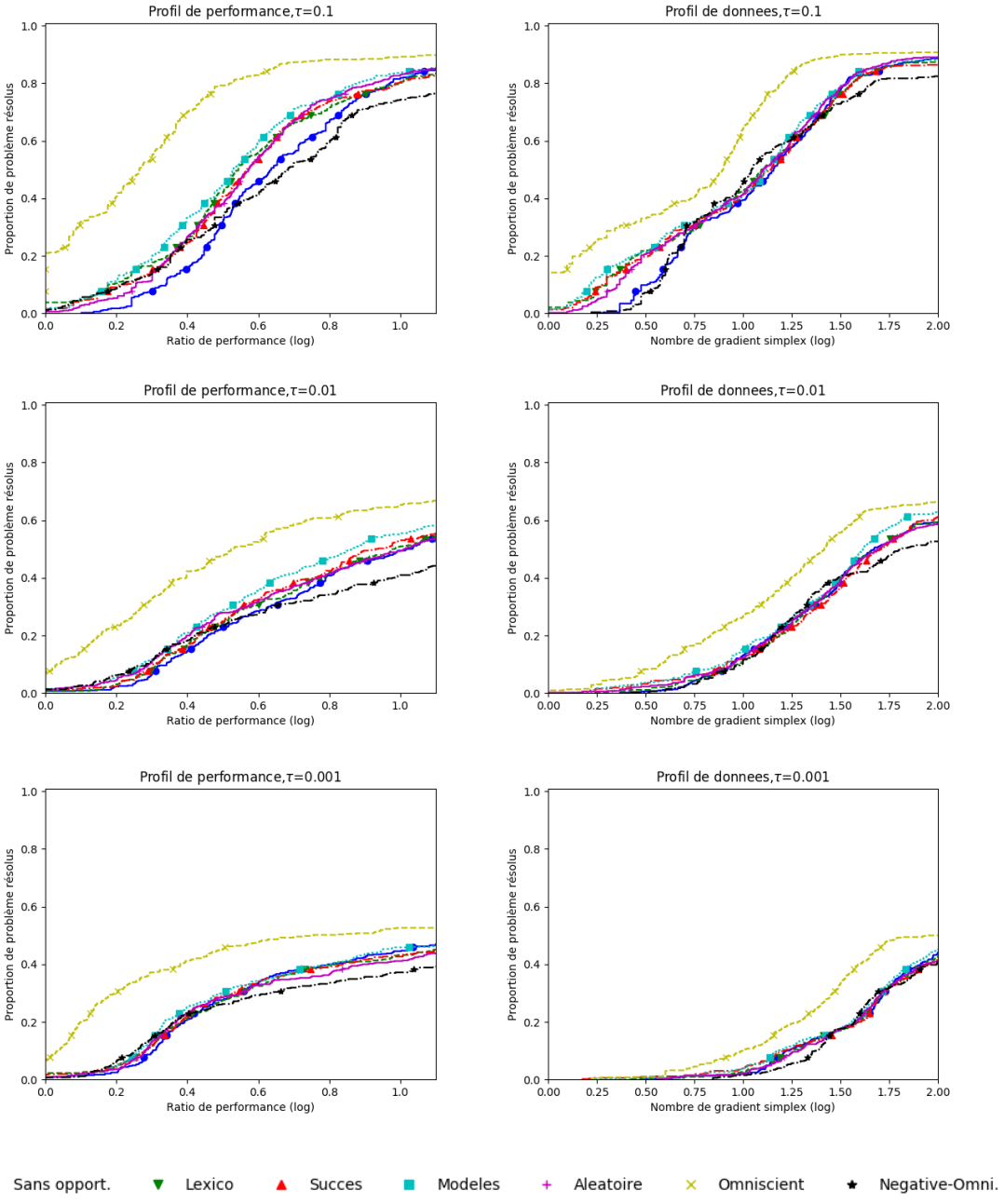

Figure 4.12 Comparaison sur problèmes non-differentiables de Moré-Wild avec MADS par défaut de NOMAD

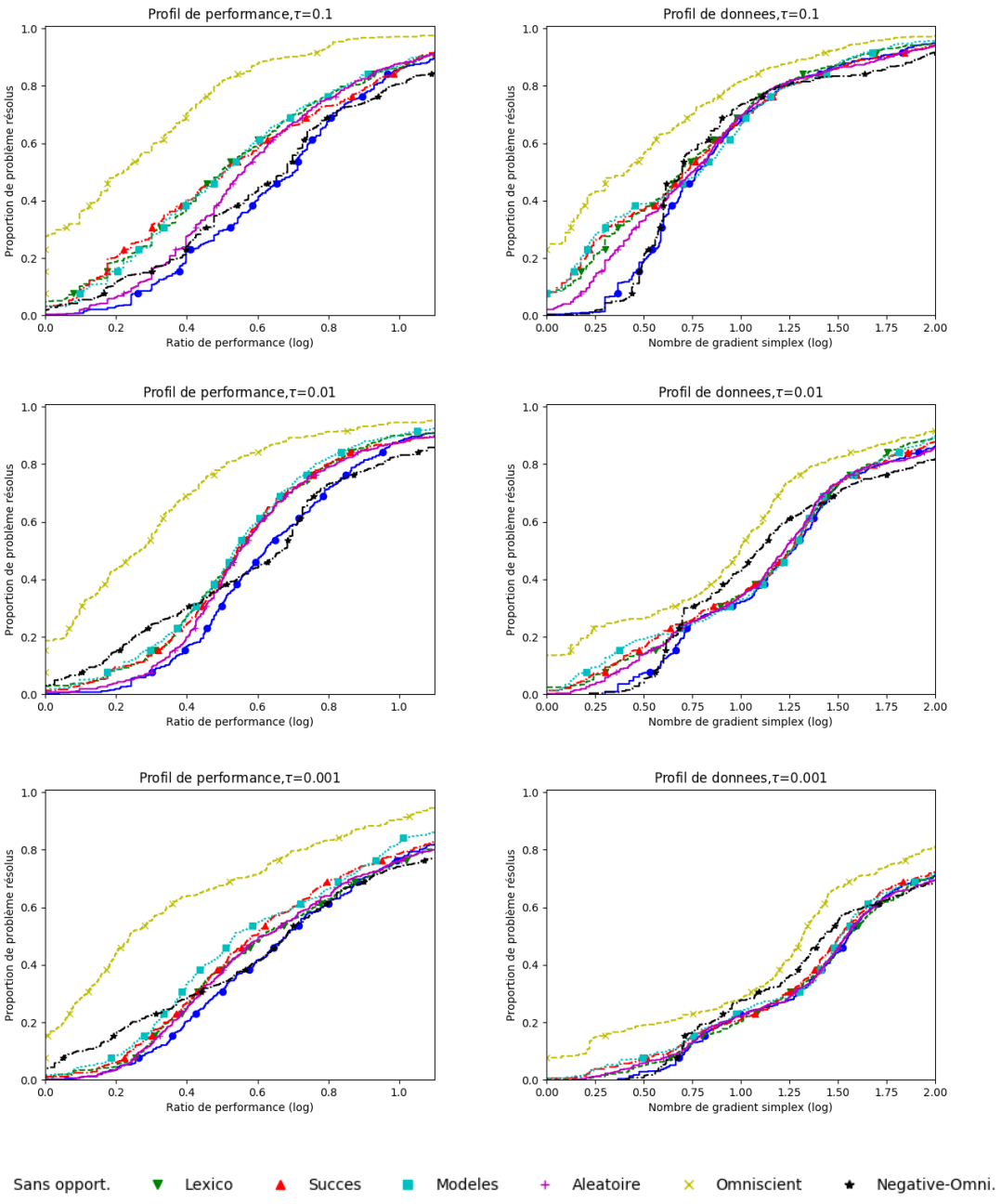

Figure 4.13 Comparaison sur problèmes bruités de Moré-Wild avec MADS par défaut de NOMAD

Commentaires Commentaires

STYRENE

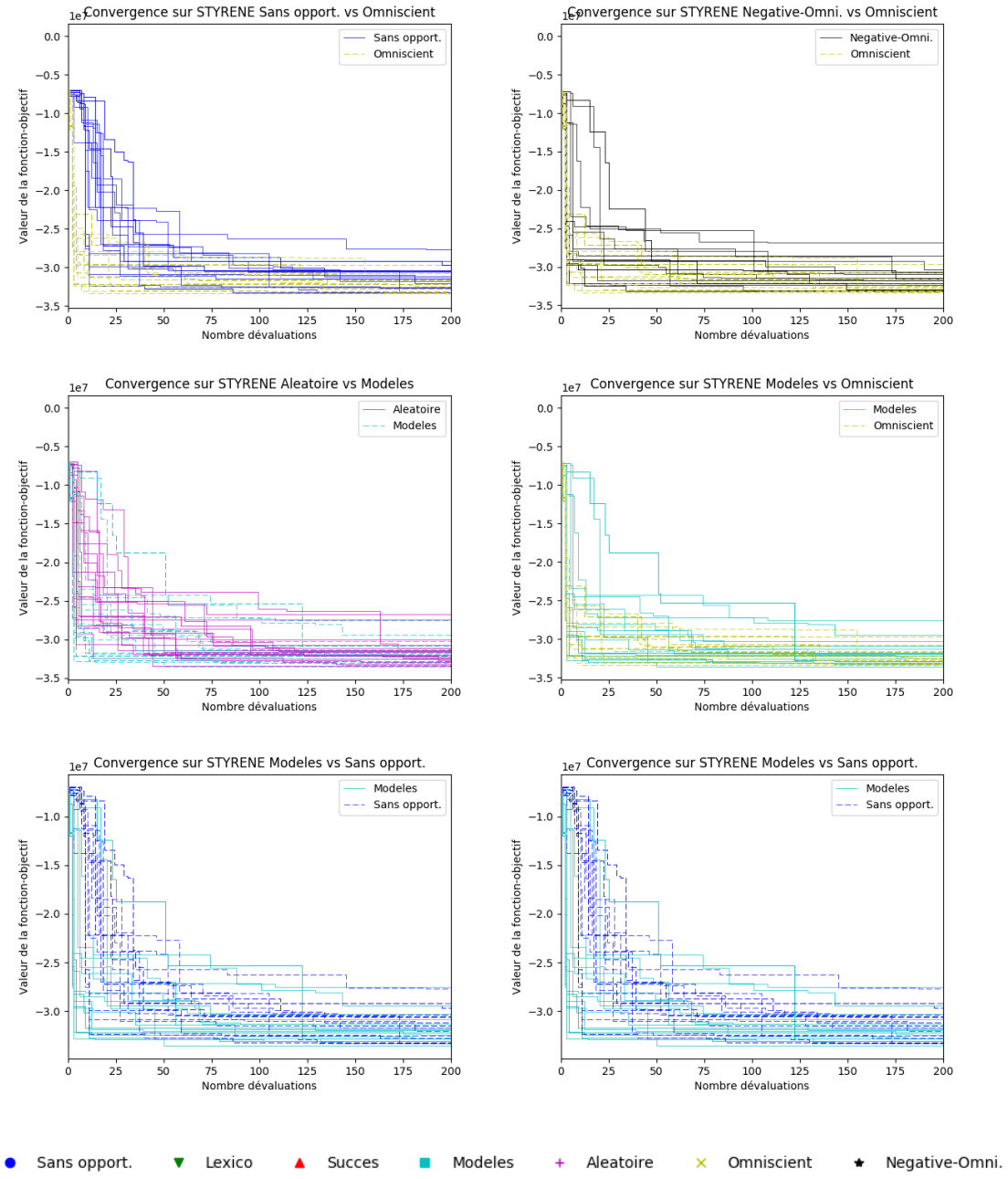

Figure 4.14 Comparaison sur STYRENE avec MADS par défaut de NOMAD

4.4.5 GSS

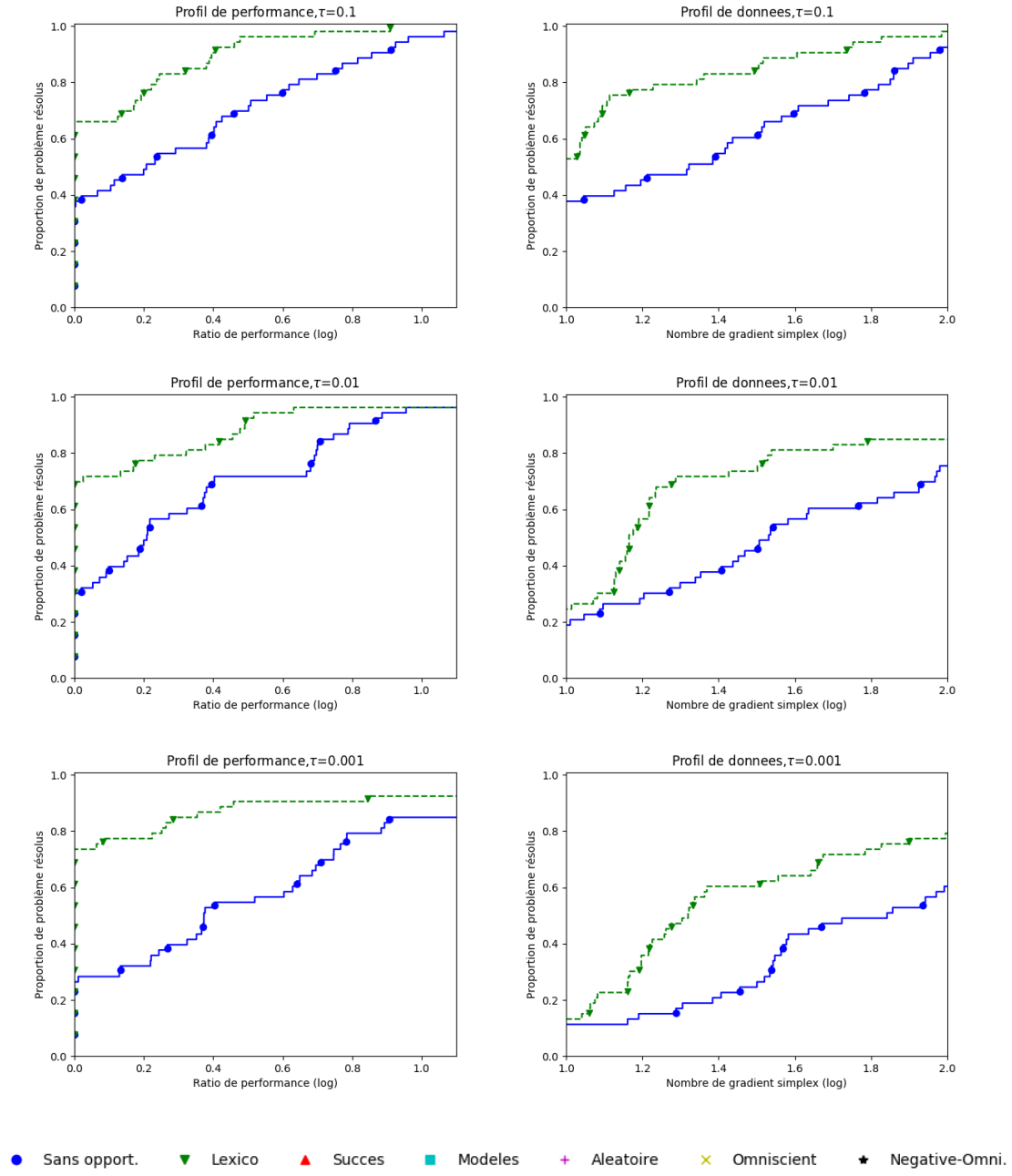

Figure 4.15 Comparaison sur problèmes lisses de Moré-Wild avec MADS

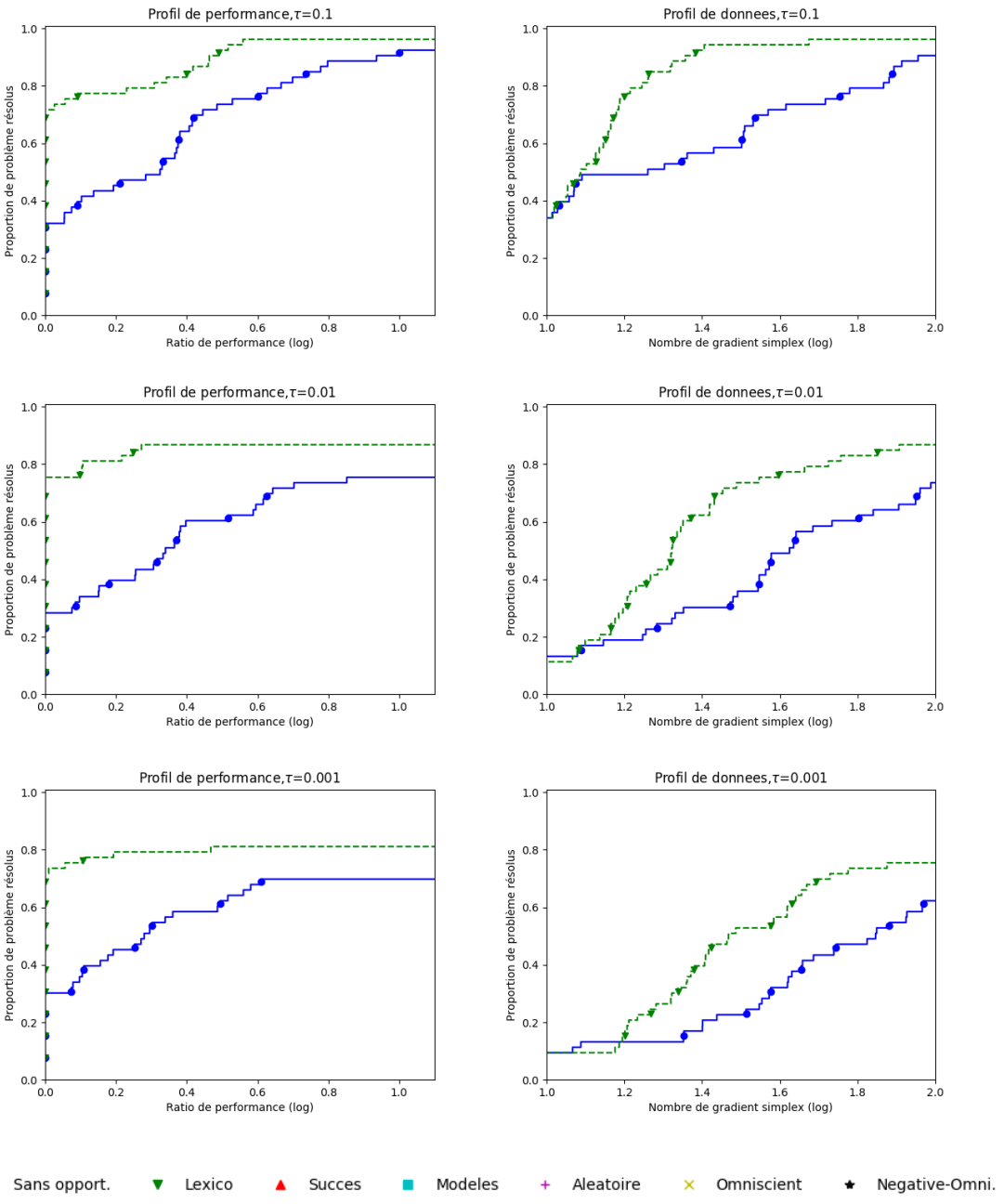

Figure 4.16 Comparaison sur problèmes non-differentiables de Moré-Wild avec MADS

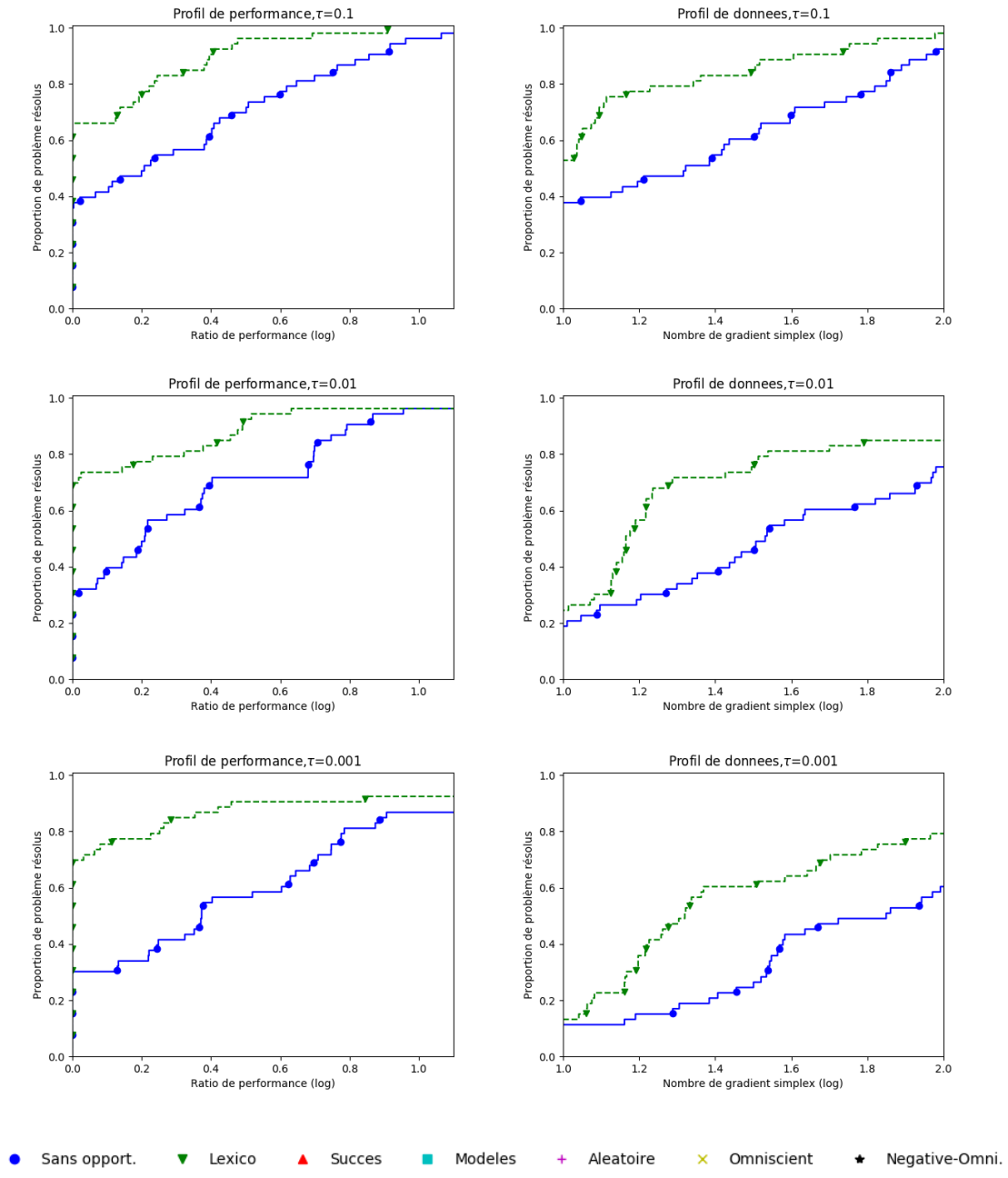

Figure 4.17 Comparaison sur problèmes bruités de Moré-Wild avec MADS

4.4.6 imfil

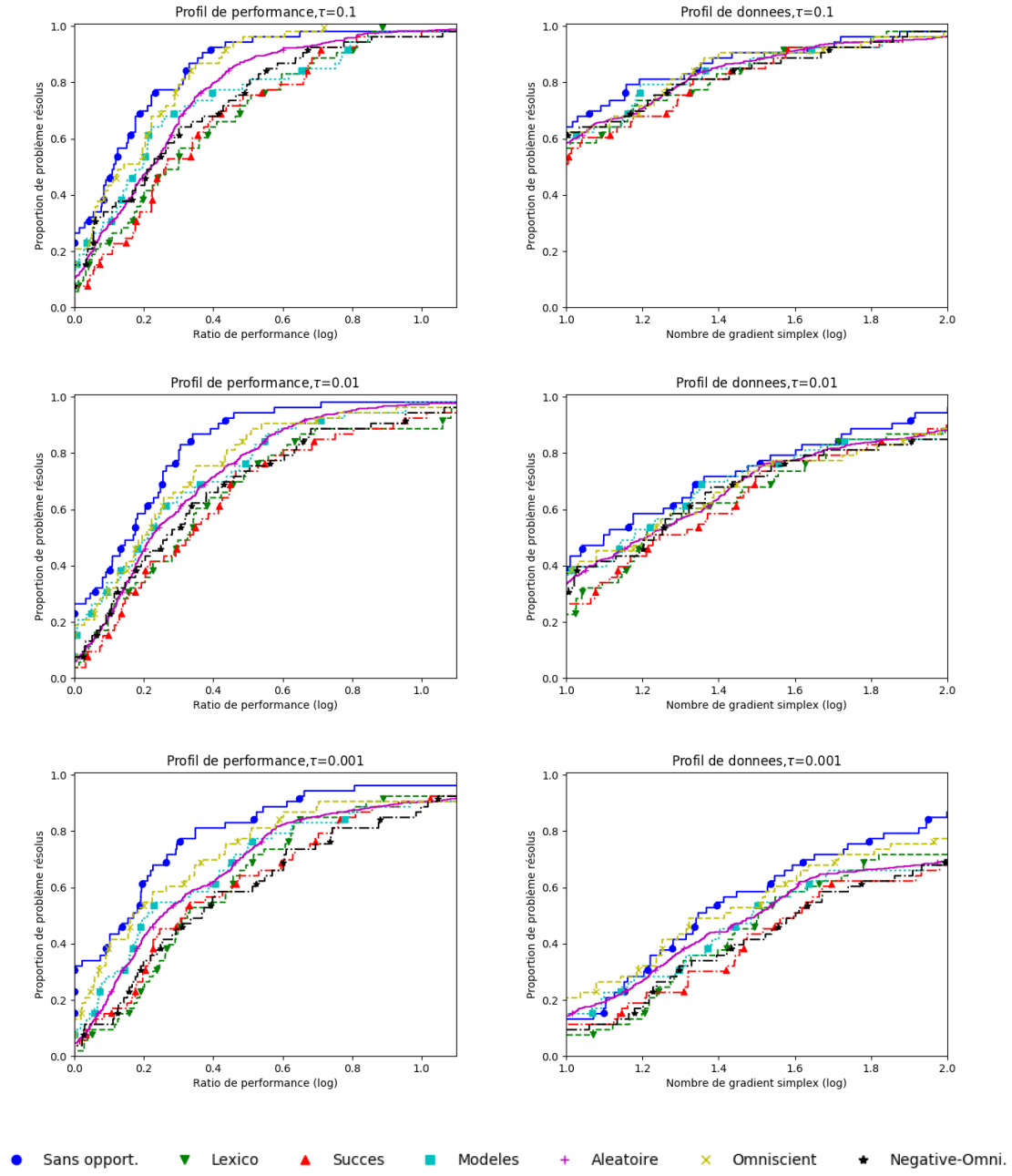

Figure 4.18 Comparaison sur problèmes lisses de Moré-Wild avec CS

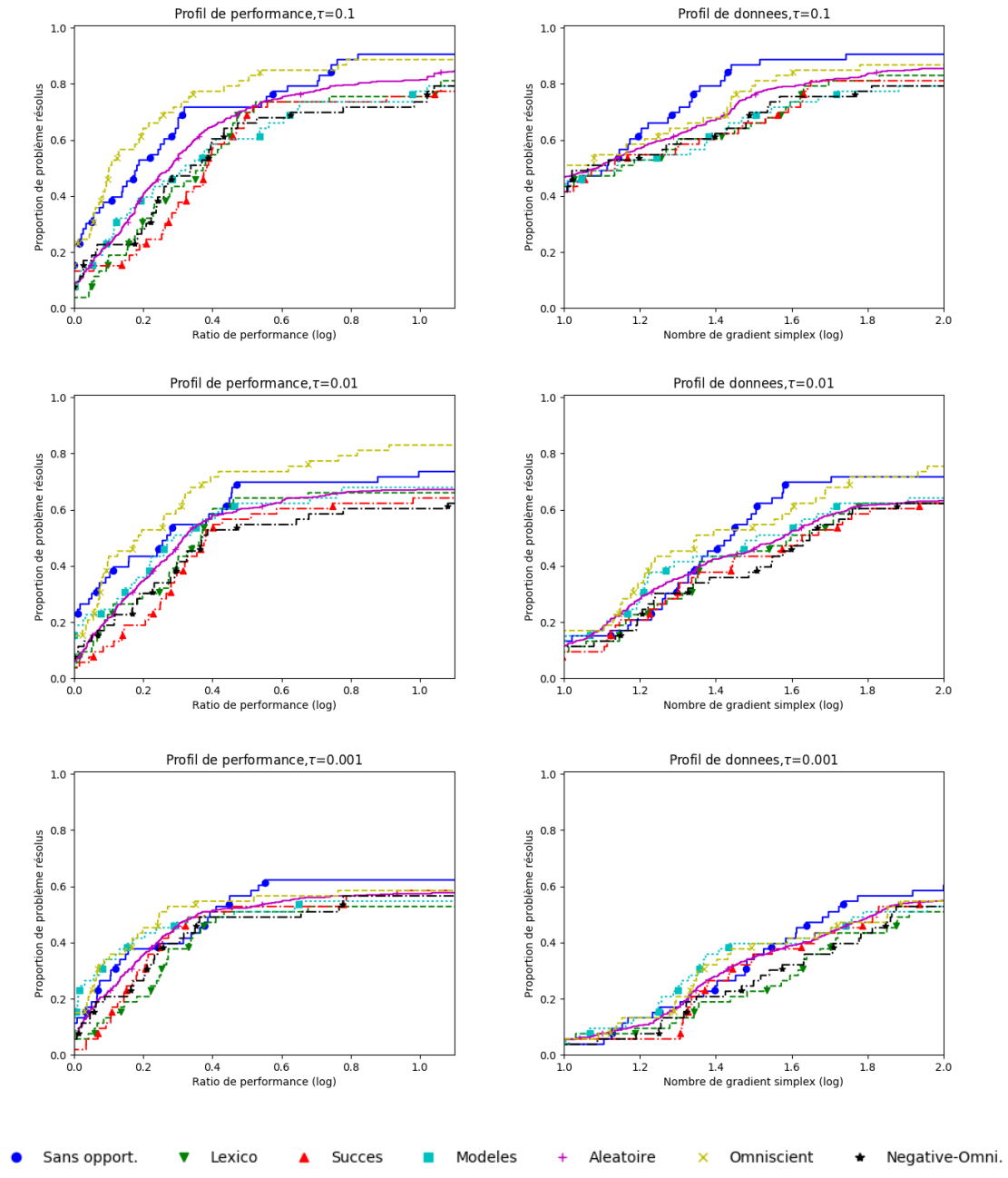

Figure 4.19 Comparaison sur problèmes non-differentiables de Moré-Wild avec CS

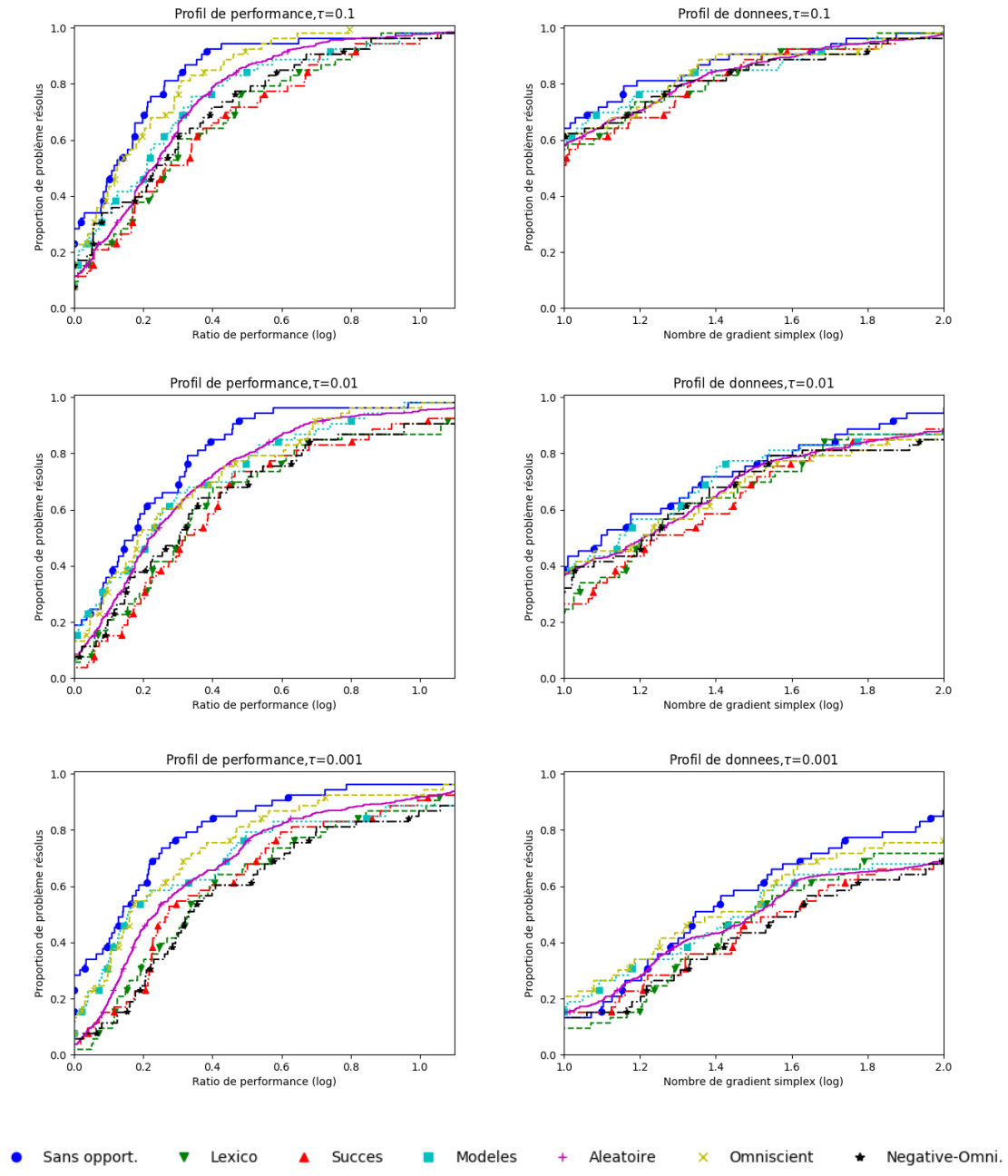

Figure 4.20 Comparaison sur problèmes bruités de Moré-Wild avec CS

Commentaires Commentaires

CHAPITRE 5 CONCLUSION

Texte.

5.1 Synthèse des travaux

— Distance à la chache (voir avec charles si on le mentionne ou si il s'agit d'un vol d'idée)
— simplex gradient, un peu comme les modèles (citer custodio)
— ordering following the original order but without reset (comme lexicographique mais sans reset, alors plus comme random) mentionné dans conn scheinberg vincente
— ordonner les points en fonction de la distance au meilleur point obtenu sur un modèle quadratique
— ordonnancement et barriere progressive

5.2 Limitations de la solution proposée

plusieurs stratégies ne sont pas applicable a toutes les implémentations

5.3 Améliorations futures

+ de stratégies. (distance cache, derivé simplexe, lexico-without reset, interaction progressive barrier-opportunisme-) parallelisme

RÉFÉRENCES

[1] M.A. Abramson, C. Audet, and J.E. Dennis, Jr. Generalized pattern searches with derivative information. *Mathematical Programming*, Series B, 100(1) :3–25, 2004.

[2] L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1) :1–3, 1966.

[3] C. Audet, C.-K. Dang, and D. Orban. Efficient use of parallelism in algorithmic parameter optimization applications. *Optimization Letters*, 7(3) :421–433, 2013.

[4] C. Audet and J.E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 14(4) :980–1010, 2004.

[5] C. Audet and J.E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1) :188–217, 2006.

[6] C. Audet and J.E. Dennis, Jr. A Progressive Barrier for Derivative-Free Nonlinear Programming. *SIAM Journal on Optimization*, 20(1) :445–472, 2009.

[7] C. Audet, J.E. Dennis, Jr., and S. Le Digabel. *Parallel Space Decomposition of the Mesh Adaptive Direct Search algorithm*, volume 5 of *The GERAD newsletters (Eleven articles published in leading journals)*, page 3. 2008.

[8] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, 2018, in press.

[9] C. Audet, A. Ianni, S. Le Digabel, and C. Tribes. Reducing the Number of Function Evaluations in Mesh Adaptive Direct Search Algorithms. *SIAM Journal on Optimization*, 24(2) :621–642, 2014.

[10] C. Audet and M. Kokkolaras. Blackbox and derivative-free optimization : theory, algorithms and applications. *Optimization and Engineering*, 17(1) :1–2, 2016.

[11] C. Audet, S. Le Digabel, and C. Tribes. NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD, 2009.

[12] I. Bongartz, A.R. Conn, N. Gould, and Ph.L. Toint. CUTE : constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1) :123–160, 1995.

[13] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. *Structural and Multidisciplinary Optimization*, 17(1) :1–13, 1999.

[14] C.G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(92) :577–593, 1965.

[15] A.R. Conn and S. Le Digabel. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1) :139–158, 2013.

[16] A.R. Conn, K. Scheinberg, and L.N. Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming*, 111(1–2) :141–172, 2008.

[17] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[18] I.D. Coope and C.J. Price. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization*, 11(4) :859–869, 2001.

[19] A.L. Custódio, J.E. Dennis, Jr., and L.N. Vicente. Using simplex gradients of nonsmooth functions in direct search methods. *IMA Journal of Numerical Analysis*, 28(4) :770–784, 2008.

[20] A.L. Custódio, H. Rocha, and L.N. Vicente. Incorporating minimum Frobenius norm models in direct search. *Computational Optimization and Applications*, 46(2) :265–278, 2010.

[21] A.L. Custódio and L.N. Vicente. Using Sampling and Simplex Derivatives in Pattern Search Methods. *SIAM Journal on Optimization*, 18(2) :537–555, 2007.

[22] C. Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76 :733–746, 1954.

[23] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2) :201–213, 2002.

[24] E. Fermi and N. Metropolis. Numerical solution of a minimum problem. Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA, 1952.

[25] R. Fletcher. Function minimization without evaluating derivatives – a review. *The Computer Journal*, 8(1) :33–41, 1965.

[26] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore and London, third edition, 1996.

[27] N. Gould and J. Scott. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software*, 43(1) :15 :1–15 :5, May 2016.

[28] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEr (and SifDec) : A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4) :373–394, 2003.

[29] I. Griva, S.G. Nash, and A. Sofer. *Linear and Nonlinear Optimization.* Society for Industrial and Applied Mathematics, 2009.

[30] R. Hooke and T.A. Jeeves. “Direct Search" Solution of Numerical and Statistical Problems. *Journal of the Association for Computing Machinery*, 8(2) :212–229, 1961.

[31] P.D. Hough, T.G. Kolda, and V. Torczon. Asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Scientific Computing*, 23(1) :134–156, 2001.

[32] C.T. Kelley. *Iterative Methods for Optimization.* Number 18 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1999.

[33] C.T. Kelley. *Implicit Filtering.* Society for Industrial and Applied Mathematics, Philadephia, PA, 2011.

[34] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search : New perspectives on some classical and modern methods. *SIAM Review*, 45(3) :385–482, 2003.

[35] R.M. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical Report 96–71, Institute for Computer Applications in Science and Engineering, Mail Stop 132C, NASA Langley Research Center, Hampton, Virginia 23681–2199, 1996.

[36] R.M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4) :1082–1099, 1999.

[37] R.M. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3) :917–941, 2000.

[38] S. Lucidi and M. Sciandrone. On the Global Convergence of Derivative-Free Methods for Unconstrained Optimization. *SIAM Journal on Optimization*, 13(1) :97–116, 2002.

[39] M. Marazzi and J. Nocedal. Wedge trust region methods for derivative free optimization. *Mathematical Programming*, 91(2) :289–305, 2002.

[40] H. D. Mittelmann. Benchmarking interior point lp/qp solvers. *Optimization Methods and Software*, 11 :655–670, 1999.

[41] J.J. Moré and S.M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1) :172–191, 2009.

[42] S.G. Nash and A. Sofer. *Linear and Nonlinear Programming.* McGraw–Hill, New York, 1996.

[43] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4) :308–313, 1965.

[44] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer, New York, 1999.

[45] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Berlin, second edition, 2006.

[46] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970. Reissued in 2000 by SIAM Publications, Philadelphia, as Vol. 30 in the series Classics in Applied Mathematics.

[47] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1) :1–25, 1997.

[48] A.I.F. Vaz and L.N. Vicente. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39(2) :197–219, 2007.