Titre: Title:	Modèles quadratiques et décomposition parallèle pour l'optimisation sans dérivées
Auteur: Author:	Nadir Amaioua
Date:	2018
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Amaioua, N. (2018). Modèles quadratiques et décomposition parallèle pour l'optimisation sans dérivées [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3186/

UNIVERSITÉ DE MONTRÉAL

MODÈLES QUADRATIQUES ET DÉCOMPOSITION PARALLÈLE POUR L'OPTIMISATION SANS DÉRIVÉES

NADIR AMAIOUA
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
JUIN 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

MODÈLES QUADRATIQUES ET DÉCOMPOSITION PARALLÈLE POUR L'OPTIMISATION SANS DÉRIVÉES

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

M. ANJOS Miguel F., Ph. D., président
M. LE DIGABEL Sébastien, Ph. D., membre et directeur de recherche
M. AUDET Charles, Ph. D., membre et codirecteur de recherche
M. CAPOROSSI Gilles, Ph. D., membre
M. DELORME Louis, Ph. D., membre externe

DÉDICACE

À mes parents

REMERCIEMENTS

Je voudrais tout d'abord remercier mes directeurs de recherche Sébastien Le Digabel et Charles Audet pour m'avoir encouragé à entreprendre cette aventure, pour tous les conseils qu'ils m'ont donnés au cours de ces dernières années et pour leurs disponibilités à m'aider en tout temps. C'était un plaisir d'effectuer mes recherches sous votre encadrement.

Un grand merci à Andy Conn dont les idées m'ont poussé à poursuivre ce doctorat et sans qui ce travail n'aurait pas été possible. Je remercie également mes collaborateurs Stéphane Alarie et Louis-Alexandre Leclaire pour m'avoir aidé et accueilli au sein de l'IREQ afin d'effectuer une partie de mes recherches. Je remercie Christophe, Viviane et Bastien qui ont toujours pu répondre à mes questions. Je tiens aussi à remercier Miguel Anjos, Gilles Caporossi, Louis Delorme et Guy Marleau d'avoir accepté de faire partie de mon jury de soutenance.

J'aimerais ensuite remercier mes parents pour avoir cru en moi et pour tout le soutien qu'ils m'ont apporté tout au long de ces années, Mariana pour m'avoir épaulé chaque jour et pousser vers l'avant, mon grand-père Omar qui m'a toujours encouragé à poursuivre mes études jusqu'au bout et ma sœur Nada qui est une source inépuisable de divertissement. Un grand merci à mes modèles de travail Mohamed Chelh, Amine Mazouz et Younes Belhabra.

Je remercie mes amis du GERAD et particulièrement Stéphane pour les discussions à propos du Foot et Mathilde qui a été la première à m'expliquer les méthodes de région de confiance lors de notre voyage aux USA. Merci à mes amis brésiliens Elisama, Michael, Stephanie, Vilmar et Alireza pour les sorties et les barbecues de ces dernières années. Un grand merci à la communauté marocaine de Montréal : Hamza, Dina, Chahid, Amine, Jalil, Mouna, Ismail, Mouad et Adham. Je tiens à remercier particulièrement mes compagnons de ma seconde vie Lifesaver, Jouraki, Orucly, Djidane et Kalyh qui ont donné un gout magique à mon doctorat.

Finalement, j'aimerais remercier tous mes professeurs qui m'ont donné envie de poursuivre mes études en mathématiques et particulièrement, M. Bouanane, M. Zoumi, M.Mahada, M. Driouche et M. Ouali.

RÉSUMÉ

L'optimisation sans dérivées (DFO) et l'optimisation de boites noires (BBO) sont deux disciplines qui traitent des problèmes dont la formulation analytique est inaccessible partiellement ou totalement et qui résultent souvent de simulations informatiques. Les algorithmes DFO et BBO s'appliquent typiquement à des problèmes de petite dimension. Parmi ces méthodes, l'algorithme de recherche directe par treillis adaptatifs (MADS) est une méthode itérative qui se base sur une discrétisation de l'espace et des directions de recherche pour sélectionner et évaluer des points de l'espace. Cette thèse explore deux extensions de MADS qui permettent d'améliorer les résultats de la méthode ainsi que de s'attaquer à des problèmes de plus grande taille.

Dans la première extension, MADS utilise des modèles dans l'étape de recherche menant à la création d'une série de sous-problèmes quadratiques avec contraintes quadratiques. Deux méthodes d'optimisation non linéaire classiques sont décrites : la fonction de pénalité exacte en norme ℓ_1 et le Lagrangien augmenté. De plus, une nouvelle méthode nommée le Lagrangien augmenté en norme ℓ_1 combine les points forts des deux algorithmes précédents. Cette dernière méthode est bien adaptée pour les problèmes quadratiques avec contraintes quadratiques vu qu'elle se base sur un terme de pénalité en norme ℓ_1 qui permet de traiter un problème quadratique par morceaux au lieu d'un problème quartique si le Lagrangien augmenté standard est utilisé.

La nouvelle méthode du Lagrangien augmenté en norme ℓ_1 est décrite pour les problèmes non linéaires avec des contraintes d'égalités. Une analyse conduite sur l'itération interne de l'algorithme prouve que la convergence vers un point stationnaire se fait avec une vitesse surperlinéaire en deux étapes. De plus, l'analyse de l'itération externe de la méthode établit que l'algorithme converge globalement et que le paramètre de pénalité est borné.

Dans la seconde extension, l'algorithme de décomposition parallèle de l'espace de la recherche directe par treillis adaptatifs (PSD-MADS), qui est une méthode parallèle asynchrone pour les problèmes de boites noires de grande taille, utilise une stratégie de sélection aléatoire des variables pour la construction des sous-problèmes. Plusieurs stratégies sont proposées pour sélectionner les variables les plus influentes du problème et explorer l'espace des solutions de manière plus efficace. Ces stratégies se basent sur des outils statistiques pour évaluer l'influence des variables sur les différentes sorties et sur la méthode de classification k-mean pour grouper les variables avec plus ou moins le même niveau d'influence. De plus, une méthode hybride qui combine cette nouvelle stratégie avec l'approche aléatoire de sélection de variables est présentée. Les nouvelles méthodes améliorent les résultats de PSD-MADS et les tests numériques sont conduits sur des problèmes de taille allant jusqu'à 4000 variables.

ABSTRACT

Derivative-free optimization (DFO) and blackbox optimization (BBO) are two fields studying problems for which the analytical formulation is partially or completely inaccessible, and which often result from computer simulations. Algorithms in DFO and BBO typically target problems with small dimension. One of these methods is the mesh adaptive direct search algorithm (MADS) which is an iterative method relying on a space discretization and search directions to select and assess new candidates. This thesis explores two extensions of the MADS algorithm that allow to improve its results and take on problems with a larger dimension.

In the first extension, MADS uses models in the search step which generates a sequence of quadratic subproblems with quadratic constraints. Two classic nonlinear optimization methods are described: the ℓ_1-exact penalty function and the augmented Lagrangian. In addition, a new method, called the ℓ_1 augmented Lagrangian, combines the strengths of both previous methods. This new approach is well suited for quadratically constrained quadratic problems (QCQP) since the ℓ_1 penalty term allows the method to optimize a piecewise quadratic problem instead of a quartic one when using the standard augmented Lagrangien.

The new ℓ_1 augmented Lagrangian is described for nonlinear problems with equality constraints. The analysis of the inner iteration of the algorithm proves a superlinear convergence to a stationary point. In addition,the analysis of the outer loop of the method establishes global convergence and shows that the penalty parameter is bounded away from zero.

In the second extension, the parallel space decomposition of the mesh adaptive direct search algorithm (PSD-MADS), which is an asynchronous parallel method for large-sized blackbox problems, uses a random selection of variables to build subproblems. Several new strategies are introduced to select the most influential variables and explore the solution space more efficiently. These strategies are based on statistical tools to quantify the influence of variables on the different outputs and use the k-mean clustering method to group variables with the same level of influence together. In addition, a hybrid method combines this new strategy with the random variable selection of the original PSD-MADS. These new methods improve the results of PSD-MADS and are tested on problems with up to 4000 variables.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

DFO	Derivative-Free Optimization
BBO	Blackbox Optimization
MADS	Mesh Adaptive Direct Search
PSD-MADS	Parallel Space Decomposition for the Mesh Adaptive Direct Search
CS	Coordinate Search
GPS	Generalized Pattern Search
GSS	Generating Set Search
APPS	Asynchronous Parallel Pattern Search
PCA	Principal Component Analysis
ANOVA	ANalysis Of VAriance
PFA	Principal Feature Analysis
ℓ_1EPF	ℓ_1 Exact Penalty Function
AUGLAG	AUGmented LAGrangian
ℓ_1AUGLAG	ℓ_1 AUGmented LAGrangian
PB	Progressive Barrier
EB	Extreme Barrier

CHAPITRE 1 INTRODUCTION

Dans plusieurs domaines, la construction d'un modèle mathématique simplifié constitue une étape importante lors de la résolution de problèmes complexes. Ce modèle mathématique identifie les paramètres et les variables de décision du problème, ainsi que les relations qui lient et contraignent ces variables et qui mesurent l'utilité et le rendement du système. De manière formelle, le modèle consiste en une fonction objectif qu'il faut minimiser ou maximiser, en plus des relations qui restreignent le fonctionnement du système, le plus souvent sous forme d'égalité ou d'inégalité, et qu'on appelle contraintes. Ce modèle devient candidat à être optimisé : c'est-à-dire que nous cherchons à trouver la configuration idéale des variables qui permet de maximiser ou minimiser l'objectif tout en respectant les contraintes du système.

Selon la nature du modèle créé, plusieurs méthodes d'optimisation peuvent être utilisées. Les techniques d'optimisation classiques se basent sur l'exploitation de la structure du problème en évaluant ces dérivées (calcul de gradients, de matrices hessiennes, etc) et posent des hypothèses spécifiques sur l'objectif et les contraintes (fonctions lisses, différentiables, etc). Par exemple, pour une fonction continûment différentiable non contrainte, un minimum local est assuré par l'intermédiaire d'un gradient nul de la fonction et d'une hessienne définie positive en ce point.

Comme les informations des dérivées sont essentielles pour cette gamme de méthodes d'optimisation, l'utilisateur doit utiliser la différentiation automatique pour obtenir les dérivées d'un problème, ou bien les coder directement à la main. Ceci n'est pas toujours possible : typiquement quand le modèle est décrit par une simulation informatique difficile à mettre sous forme mathématique exacte ou bien quand l'accès est limité à des exécutables binaires pour des logiciels ou des codes privés.

Il est possible aussi d'utiliser une méthode de différences finies pour estimer les dérivées. Cependant, quand les fonctions sont bruitées ou bien quand les évaluations de ces fonctions sont très coûteuses, le calcul de différences finies devient problématique.

Beaucoup de modèles qui décrivent des problèmes réels présentent des structures où les dérivées d'un problème sont difficiles à évaluer ou ne sont pas fiables. Dans ce cas, on préfère une méthode d'optimisation sans dérivées (DFO) ou une approche d'optimisation de boites noires (BBO). Tout comme dans [22], nous distinguons l'optimisation sans dérivées qui opère sur des problèmes pour lesquels les dérivées existent, mais impossible d'accès et l'optimisation de boites noires qui traite des problèmes qui n'admettent pas de dérivées.

Une boite noire (voir la figure 1.1) est typiquement un code informatique qui pour une ou plusieurs

entrées renvoie une sortie (ou plusieurs sorties dépendant du type de problème).

Figure 1.1 Illustration d'une boite noire.

Depuis les années cinquante, plusieurs méthodes et heuristiques ont été implémentées pour résoudre les problèmes de type boite noire. Parmi ces approches, nous comptons principalement les méthodes de recherche directe qui sont accompagnées par des résultats de convergence pour valider leurs approches. Nous nous intéressons particulièrement à l'algorithme MADS [18] dans cette thèse qui est conçu pour l'optimisation de boites noires. Une extension de MADS, décrite dans [49], permet d'utiliser des modèles quadratiques pour effectuer des interpolations du problème considéré et interagir avec des dérivées pour guider la recherche. Ceci montre que MADS est également adéquat pour optimiser des problèmes de type DFO.

L'algorithme MADS se base sur la comparaison des valeurs de l'objectif f et l'évaluation des contraintes c (voir la figure 1.2) pour rejeter ou retenir le point courant sans utiliser aucune information de dérivée. Les fonctions f et c sont définies plus en détail dans l'équation (2.1).

NOMAD [109] est un logiciel d'optimisation de boites noires qui implémente l'algorithme MADS. Toute amélioration ou innovation apportées à l'algorithme MADS impacte directement sur le déroulement de NOMAD et permet de fournir de meilleurs résultats plus rapidement. Par exemple, pour l'extension d'utilisation des modèles quadratiques mentionnée précédemment, la résolution des sous-problèmes est conduite par une nouvelle instance simplifiée de MADS. Cette approche ne semble pas adéquate vu que MADS n'utilise pas les dérivées pour exploiter la structure d'un problème quadratique avec contraintes quadratiques.

Une deuxième problématique considérée dans cette thèse concerne l'algorithme PSD-MADS [20] qui est une parallélisation asynchrone de l'algorithme MADS. Cet algorithme est conçu dans le but de traiter des problèmes de plus grande dimension que ce qui est considéré habituellement dans

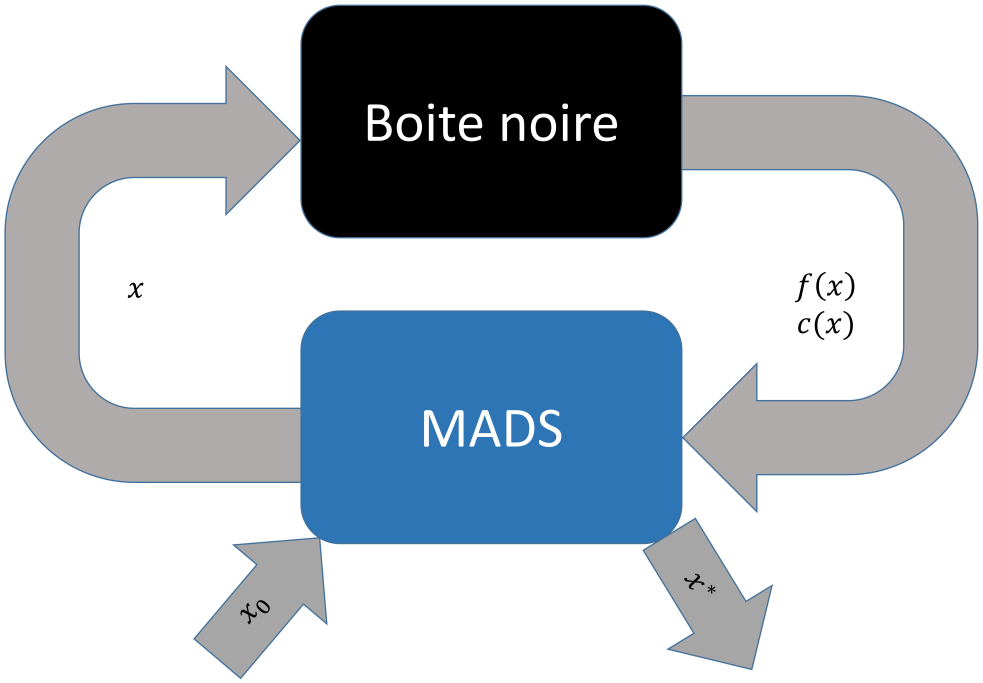


Figure 1.2 Interaction de MADS avec une boite noire.

les contextes DFO et BBO. PSD-MADS se base sur la décomposition en espace parallèle via une sélection aléatoire des variables pour construire des sous-problèmes qui sont traités en parallèle.

Cette thèse a pour objectif d'améliorer ces deux extensions de MADS pour pouvoir résoudre les problèmes de manière plus rapide et plus efficace et de s'attaquer à des instances de taille plus importantes. Pour la première extension, nous proposons des options plus appropriées que MADS pour la résolution des sous-problèmes quadratiques dans la phase de recherche de MADS. Ceci permet de définir une nouvelle approche d'optimisation non linéaire appelée le Lagrangien augmenté en norme ℓ_1 est qui est adéquate pour les problèmes quadratiques avec contraintes quadratiques. La deuxième extension se concentre sur l'algorithme de PSD-MADS et propose une nouvelle approche pour sélectionner les variables les plus influentes d'un problème via une étude statistique.

Nous commençons par présenter une revue de littérature dans le chapitre 2 sur les méthodes de recherches directes, l'utilisation des modèles quadratiques dans MADS, les algorithmes parallèles en DFO ainsi que des techniques de sélection de variables. Le chapitre 3 détaille l'organisation des trois thèmes traités dans cette thèse. Dans le chapitre 4, nous nous concentrons sur l'amélioration de la résolution des sous-problèmes quadratiques dans l'algorithme MADS. Le chapitre 5 reprend la méthode du Lagrangien augmenté en norme ℓ_1 introduite dans le chapitre 4 et présente des résultats de convergence pour cette nouvelle approche. Le troisième thème, portant sur l'exploration

de nouvelles techniques de sélection de variables pour la construction de sous-problèmes dans l'algorithme PSD-MADS, est décrit dans le chapitre 6. Finalement, une conclusion fait la synthèse de la recherche effectuée et introduit une discussion sur les perspectives des travaux réalisés.

CHAPITRE 2 REVUE DE LITTÉRATURE

Cette partie présente l'univers de l'optimisation sans dérivées (DFO) et de l'optimisation de boites noires (BBO). Elle décrit des méthodes de traitement de contraintes, des méthodes de recherche directe, ainsi que l'utilisation de modèles quadratiques dans l'algorithme MADS. Cette section présente aussi un sommaire de plusieurs méthodes utilisées dans la littérature pour traiter les sous-problèmes quadratiques, des méthodes de décomposition parallèle de l'espace, du parallélisme en DFO et des méthodes de sélection de variables.

Soit le problème d'optimisation avec contraintes suivant :

$$\min_{x \in \Omega} \quad f(x), \tag{2.1}$$

avec :

$$\Omega = \{x \in \mathcal{X} : c_j(x) \leq 0, j \in \{1, 2, \ldots, m\}\}, \tag{2.2}$$

où m est un entier positif, $\mathcal{X} \subseteq \mathbb{R}^n$, f et c_j, $j \in \{1, 2, \ldots, m\}$, sont des fonctions réelles.

2.1 Optimisation sans dérivées et optimisation de boites noires

S'il est possible de calculer facilement ou d'accéder aux dérivées d'un problème d'optimisation, il est toujours conseillé d'utiliser des méthodes basées sur l'évaluation des gradients. Quand ces informations de gradient ne sont pas disponibles, il faut utiliser une méthode qui ne se base pas sur les dérivées [22, 51].

L'optimisation sans dérivées (DFO) est le domaine de l'étude des algorithmes d'optimisation qui n'utilisent pas les dérivées. Typiquement, l'optimisation sans dérivées traite des problèmes produits par des simulations informatiques. Ces problèmes peuvent être différentiables, mais l'accès aux dérivées n'est pas disponible et leurs approximations sont coûteuses ou ne sont pas fiables.

Il faut distinguer DFO de l'optimisation de boite noire (BBO) qui, comme son nom l'indique, traite des boites noires. Une boite noire est un problème qui pour chaque entrée renvoie une sortie sans avoir de connaissances sur le moyen dont cette sortie est obtenue. En BBO, les boites noires peuvent ne pas être des fonctions puisqu'elles peuvent renvoyer deux valeurs différentes pour une même entrée. C'est le cas, par exemple, d'une boite noire qui contient un processus ou un bruit stochastique.

En général, les problèmes en DFO sont traités par des méthodes qui sont supportées par des analyses de convergence. Parmi ces méthodes, nous pouvons citer les approches basées sur les modèles. Ces méthodes opèrent par construction d'approximations des fonctions f et c (améliorable par utilisation d'une recherche linéaire ou d'une région de confiance) et qui doivent prendre en compte la structure géométrique du problème. Ces méthodes sont moins efficaces sur les problèmes bruités ou qui présentent des contraintes cachées. De plus, elles sont difficiles à paralléliser. Beaucoup de méthodes reposent sur cette approche [50, 140]. Dans le même esprit, quelques méthodes tentent d'approximer les gradients des fonctions comme dans [54].

Par contre, les problèmes en BBO peuvent être traités par des méthodes de recherche directe qui sont supportées par une analyse de convergence. Ces approches se basent uniquement sur la comparaison des valeurs de l'objectif f et l'évaluation des contraintes c pour rejeter ou retenir le point courant. De nombreux algorithmes utilisent ces approches tels que l'algorithme de Nelder-Mead [135, 141], l'algorithme de Hooke et Jeeves [95], la recherche par coordonnées (CS) [60], la recherche généralisée par motifs (GPS) [16, 161], l'algorithme de recherche directe par treillis adaptatif (MADS) [18] et l'algorithme de recherche par génération d'ensembles (GSS) [107]. Il est possible également d'optimiser les problèmes en BBO en utilisant des heuristiques qui, généralement, ne sont accompagnées d'aucun résultat de convergence.

Les applications des méthodes BBO et DFO apparaissent dans une multitude de domaines telles que la géométrie moléculaire [121, 8], la reconnaissance faciale [37], la conception d'un rotor d'hélicoptère [31, 151], l'optimisation de la production pétrolière [58, 75], la gestion des fronts incendies de forêt [102] et des ressources hydrauliques [7, 67, 127, 133], la conception des procédés et des alliages [73, 74, 145] et d'un bouclier thermique [1, 105], la recherche clinique pour tester de nouveaux médicaments [165] et l'optimisation des dimensions d'une valve cardiovasculaire pour les enfants atteints de malformations cardiaques congénitales [120, 167]. (voir [12] pour plus d'applications)

2.2 Traitement des contraintes

Dans cette partie, une taxonomie des contraintes et deux méthodes de traitement des contraintes sont présentées.

2.2.1 Types de contraintes

Une taxonomie des contraintes en DFO et BBO est introduite dans [110]. Elle propose de classifier les contraintes suivant quatre critères : une contrainte peut être quantifiable ou non quantifiable, relaxable ou non relaxable, algébrique ou résultant d'une simulation et finalement connue ou cachée.

Tout d'abord, une contrainte est dite quantifiable si le degré de réalisabilité ou de violation peut être quantifié. Par opposition, une contrainte non quantifiable ne dévoile que si elle est réalisable ou pas en un point donné sans donner d'information sur le degré de faisabilité ou de violation. Les contraintes c_j, $j \in \{1, 2, \ldots, m\}$, du problème (2.1) sont toujours quantifiables. Si un problème dispose d'une contrainte non quantifiable, celle-ci fait partie de l'ensemble $\mathcal{X}$. À noter que l'ensemble $\mathcal{X}$ peut se composer de contraintes quantifiables et non quantifiables.

Ensuite, une contrainte est relaxable s'il n'est pas obligatoire de la satisfaire pour obtenir des valeurs de sortie fiables. Pour le problème (2.1), les contraintes c_j, $j \in \{1, 2, \ldots, m\}$, sont relaxables. Cependant, les contraintes représentées par l'ensemble $\mathcal{X}$ ne sont pas relaxables car un point $x \notin \mathcal{X}$ est rejeté immédiatement. La plupart du temps, l'ensemble $\mathcal{X}$ est souvent défini par des contraintes de bornes ou des contraintes linéaires.

De plus, une contrainte est dite algébrique s'il est possible de confirmer sa réalisabilité sans exécuter une simulation. Pour notre problème (2.1), les contraintes c_j, $j \in \{1, 2, \ldots, m\}$, souvent résultent de simulations.

Finalement, une contrainte est connue si elle est donnée explicitement dans la formulation du problème. Dans le cas contraire, la contrainte est dite cachée. Toutes les contraintes décrites dans l'ensemble Ω (2.2) sont des contraintes connues. Les contraintes cachées surviennent le plus souvent lors de l'évaluation d'un point $x \in \Omega$ qui semble réalisable à priori, mais que cette simulation échoue pour une quelconque raison. Une contrainte cachée est, par conséquent, une contrainte non relaxable, non quantifiable et résulte d'une simulation.

Pour revenir à notre comparaison précédente entre DFO et BBO, un problème DFO est composé par des contraintes connues et quantifiables alors qu'un problème BBO contient au moins, soit une contrainte non quantifiable ou cachée, soit une sortie qui ne peut pas être interpolée par une fonction lisse.

Dans la suite, nous évoquons deux méthodes de traitement des contraintes dans le contexte sans dérivées. Nous omettons de décrire la méthode du filtre [17, 64] qui traite les contraintes d'inégalités puisque nous présentons la méthode de la barrière progressive [19] qui est une évolution de l'algorithme du filtre.

2.2.2 Barrière extrême

La méthode de la barrière extrême (EB) [18] se base sur le problème d'optimisation (2.1) pour définir la fonction suivante :

$$f_\Omega(x) = \begin{cases} f(x) & \text{si } x \in \Omega, \\ +\infty & \text{si } x \notin \Omega. \end{cases} \tag{2.3}$$

Donc, au lieu de travailler sur le problème contraint (2.1), une méthode qui adopte la barrière extrême pour traiter les contraintes, va plutôt se concentrer sur le problème suivant :

$$\min_{x \in \mathcal{X}} \quad f_\Omega(x) \tag{2.4}$$

Cette méthode peut être utilisée avec tout type de contraintes, y compris les contraintes non quantifiables.

2.2.3 Barrière progressive

La méthode de la barrière progressive (PB) est utilisée pour traiter les contraintes relaxables. Pour utiliser cet algorithme, nous devons définir la fonction de violation des contraintes [64] et la notion de domination des points dans le cadre de la PB. Pour essayer de trouver un point réalisable, cette approche utilise, tout d'abord, le degré de violation des contraintes via la fonction :

$$h(x) = \begin{cases} \sum_{j=1}^{m} (\max\{c_j(x), 0\})^2 & \text{si } x \in \mathcal{X}, \\ +\infty & \text{si } x \notin \mathcal{X}. \end{cases} \tag{2.5}$$

Cette fonction est appelée fonction de violation des contraintes et retourne la valeur 0 si toutes les contraintes sont satisfaites (i.e. $x \in \Omega$), la valeur infinie si une des contraintes non relaxables est violée (i.e. $x \notin \mathcal{X}$) ou bien la valeur de la somme des carrés de violations des contraintes relaxables si toutes les contraintes non relaxables sont satisfaites (i.e. $x \in \mathcal{X}$ et $x \notin \Omega$).

Ensuite, il faut définir la notion de domination des points pour l'algorithme de barrière progressive.

Dans le cas sans contraintes, pour comparer deux points, il suffit de comparer leurs évaluations par la fonction objectif. Dans le même esprit, un point réalisable $x^{feas} \in \Omega$ est dominé par un autre point réalisable $y^{feas} \in \Omega$ si $f(x^{feas}) < f(y^{feas})$. On note : $x^{feas} \prec_f y^{feas}$. Par contre, pour le cas non réalisable, il est possible de comparer deux points soit en utilisant la fonction objectif ou bien en utilisant la fonction de violation des contraintes. Du coup, pour deux points non réalisables x^{inf} et y^{inf} dans $\mathcal{X} \setminus \Omega$, x^{inf} est dit dominé par y^{inf} dans l'un des deux cas suivants :

- $f(x^{inf}) < f(y^{inf})$ et $h(x^{inf}) \leq h(y^{inf})$
- $f(x^{inf}) \leq f(y^{inf})$ et $h(x^{inf}) < h(y^{inf})$

et on note : $x^{inf} \prec_h y^{inf}$.

Cette distinction dans la définition entre point réalisable et non réalisable est due au fait que l'algorithme de barrière progressive utilise deux itérés et non pas un seul : un itéré réalisable $x^{feas} \in \Omega$ qui représente le point avec la meilleure valeur de l'objectif et un itéré non réalisable $x^{inf} \in \mathcal{X} \setminus \Omega$ qui est un point non dominé avec la meilleure valeur de l'objectif et avec $h(x^{inf})$ inférieur à une barrière limite $h^{max} \in \mathbb{R}_+$ qui décroit progressivement. L'algorithme 1 donne une vue générale de la méthode de la barrière progressive.

Algorithme 1: L'algorithme de barrière progressive (PB)

1-Initialisations
$\Delta > 0, h^{max} = +\infty$
x^{feas} et x^{inf} sont initialisés à partir de x_0
2-Arrêt
Mise à jour x^{feas} et x^{inf}
Si une condition d'arrêt est satisfaite : Stop. Sinon : aller à **3**
3-Recherche Globale (Optionnelle)
Utilisation d'une méthode de recherche globale (heuristique, approche basée sur des modèles, définie par l'utilisateur, etc) pour générer une liste finie de points à évaluer $\mathcal{S}$
Si $x^{test} \prec_f x^{feas}$ ou $x^{test} \prec_h x^{inf}$ pour $x^{test} \in \mathcal{S}$
augmenter Δ
$h^{max} \leftarrow h(x^{inf})$
aller à **2**
Sinon
aller à **4**
fin Si
4-Sonde locale
Utilisation d'une méthode DFO/BBO pour générer une liste $\mathcal{P}$ de point aux alentours des itérés réalisable x^{feas} et non réalisable x^{inf} dans un rayon Δ
Si $x^{test} \in \mathcal{P}$ existe tel que $x^{test} \prec_f x^{feas}$ ou $x^{test} \prec_h x^{inf}$
augmenter Δ
$h^{max} \leftarrow h(x^{inf})$
Sinon
Si $0 < h(x^{test}) < h(x^{inf})$ pour un point $x^{test} \in V$ (V est l'ensemble de points évalués précédemment)
$h^{max} \leftarrow \max\{h(x^{test})$ tel que $h(x^{test}) < h(x^{inf}), x^{test} \in V\}$
Sinon
diminuer Δ
$h^{max} \leftarrow h(x^{inf})$
fin Si
fin Si
aller à **2**

2.3 Méthodes de recherche directe

L'algorithme MADS est une généralisation de l'algorithme GPS, qui est lui même inspiré de la méthode CS. Cette section donne une description pour chacune de ces trois approches.

2.3.1 Recherche par coordonnées

Pour cette méthode on considère le problème sans contraintes suivant :

$$\min_{x \in \mathbb{R}^n} \quad f(x), \tag{2.6}$$

où $f : \mathbb{R}^n \to \mathbb{R}$. L'algorithme de recherche par coordonnées (CS) [60] est un algorithme itératif qui utilise une discrétisation spécifique pour explorer l'espace. À l'itération k, l'algorithme CS va évaluer f sur $2n$ points de sonde qui se situent dans les directions de base de $\mathbb{R}^n$ à une distance Δ_k de l'itéré courant x_k appelé centre de sonde. Δ_k est appelé paramètre du treillis et régit la discrétisation de l'espace à l'itération k. L'ensemble des points de sonde (appelé cadre) est défini alors de la manière suivante :

$$P_k = \{x_k \pm \Delta_k e_i : i = 1, 2, \ldots, n\} \tag{2.7}$$

avec e_i la $i^{\text{ème}}$ colonne de la matrice identité I_n. Il est possible de donner une vue générale de l'algorithme CS :

La condition d'arrêt de l'algorithme peut prendre plusieurs formes : atteindre un nombre maximum d'évaluations de la fonction objectif, définir une valeur limite sur le paramètre du treillis Δ, dépasser un temps d'exécution prédéfini, pas d'amélioration après un nombre d'évaluations prédéfini, atteindre une valeur limite de la fonction objectif, etc.

Dans l'algorithme CS, la stratégie par défaut est d'évaluer tous les points de l'ensemble de sonde P_k et si le meilleur point de cet ensemble à une valeur objectif inférieur à celle de l'itéré courant alors il devient le nouvel itéré. Il est possible d'améliorer l'algorithme en adoptant une stratégie différente : l'algorithme sélectionne le premier point de l'ensemble de sonde qui présente une amélioration par rapport à l'itéré courant et évite d'évaluer le reste de la liste des points. Cette stratégie est nommée la stratégie opportuniste et, avec elle, l'ordre d'évaluation des points devient très important (voir [38] pour plus de détails sur l'opportunisme en DFO). Une possibilité est de réordonner les points de P_k de façon à toujours commencer par la direction qui a assuré le dernier succès.

Algorithme 2: L'algorithme de recherche par coordonnées

1-Initialisations
$k = 0, x_k, \Delta_k$
2-Arrêt
Si une condition d'arrêt est satisfaite : Stop. Sinon : aller à **3**
3-Itération k
Évaluation de f sur l'ensemble de sonde P_k
4-Mises à jour
Si un meilleur point $x \in P_k$ est trouvé ; $f(x) < f(x_k)$
$x_{k+1} \leftarrow x$
$\Delta_{k+1} \leftarrow \Delta_k$
Sinon
$x_{k+1} \leftarrow x_k$
$\Delta_{k+1} \leftarrow \frac{\Delta_k}{2}$
fin Si
$k \leftarrow k + 1$
aller à **2**

L'algorithme de recherche par coordonnées est supporté par quelques résultats de convergence. Soit une fonction f continue bornée et soit une suite de points $\{x_k\}$ où l'algorithme CS échoue et qui converge vers $\hat{x}$. Pour toute direction $d \in \{\pm e_k : k \in \{1, 2, \ldots, m\}\}$ pour laquelle $f'(\hat{x}; d)$ existe, $f'(\hat{x}; d) \geq 0$. De plus, si f est continûment différentiable, alors :

$$\nabla f(\hat{x}) = 0. \tag{2.8}$$

Comme les directions de recherche sont très limitées dans cet algorithme, CS n'est pas très efficace. Un exemple critique est décrit dans [22]. Soit la fonction convexe :

$$\begin{array}{rccc} f: & \mathbb{R}^2 & \mapsto & \mathbb{R} \\ & x & \rightarrow & \|x\|_\infty \end{array} \tag{2.9}$$

et soit le point de départ $x^0 = [1, 1]^\top$. La figure 2.1 illustre cette situation en mettant en avant la courbe de niveau $f(x) = 1$, le point x^0 ainsi que les directions de recherche de l'algorithme CS.

On remarque que pour les directions de bases e_1 et e_2 la valeur de f se détériore et pour les directions $-e_1$ et $-e_2$, f reste constante. CS reste donc bloqué au point de départ.

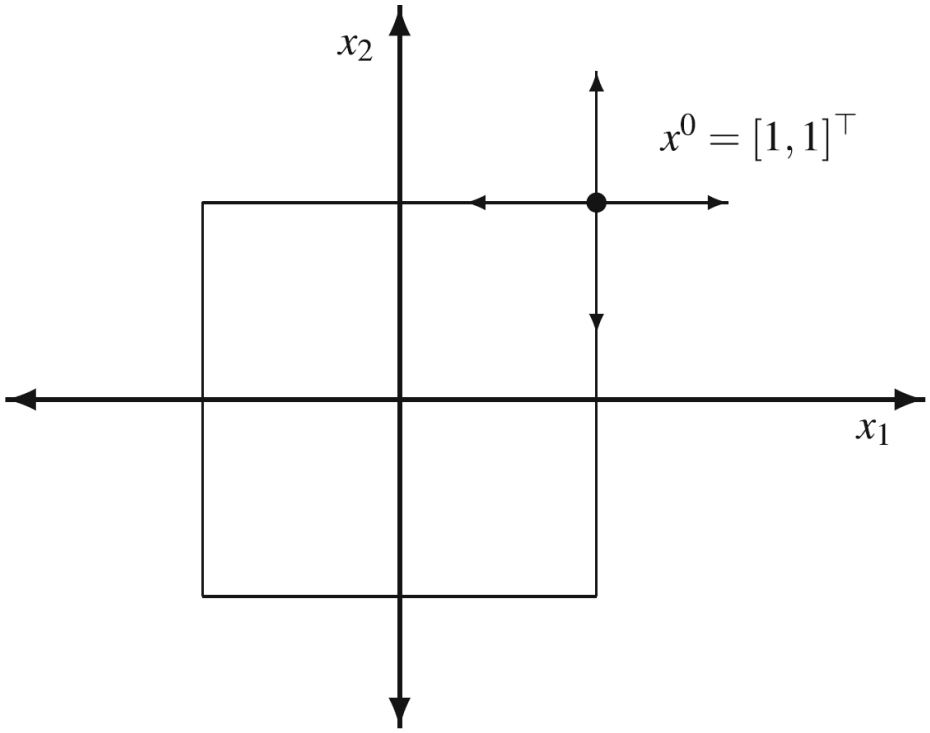


Figure 2.1 Exemple où l'algorithme CS échoue (figure adaptée de [22]).

2.3.2 Recherche par motifs généralisée

L'algorithme de recherche généralisée par motifs (GPS) est présenté pour la première fois par Torczon [161] dans le cadre de l'optimisation non linéaire sans contraintes. GPS est, plus tard, étendu aux problèmes avec des contraintes de bornes [111], avec des contraintes linéaires [112], avec des variables mixtes [15] puis avec des contraintes non linéaires [17, 113].

L'algorithme GPS constitue une évolution de l'algorithme CS en introduisant deux améliorations très importantes :

- GPS ajoute une étape de recherche globale flexible avant la sonde locale. Cette recherche doit générer un nombre fini de points à évaluer.
- La définition des directions de recherche est plus flexible dans GPS. De plus, le nombre de directions à considérer augmente considérablement.

Le choix de l'ensemble des directions possible D est régi par quelques règles dont l'intérêt est décrit dans [11]. À chaque itération k, la sonde locale commence par sélectionner un sous-ensemble $D_k \subset D$: D_k doit constituer un ensemble générateur positif [55]. GPS opère sur un treillis défini par :

$$M_k = \{x + \Delta_k Dz : x \in V_k, z \in \mathbb{N}^p\} \tag{2.10}$$

où V_k est l'ensemble des points déjà visités au début de l'itération k et Δ_k est appelé paramètre de taille de treillis. Grâce à cette définition, tous les points évalués doivent se situer sur le treillis (peu importe que les points soient évalués en phase de recherche globale ou pendant la sonde locale). La mise à jour du paramètre du treillis se fait suivant la formule suivante :

$$\Delta_{k+1} \leftarrow \tau^{\omega_k} \Delta_k \tag{2.11}$$

où $\tau \in \mathbb{Q}$ et ω_k est un entier positif si un nouvel itéré est trouvé à l'itération k. Sinon ω_k doit être négatif. Enfin, il faut redéfinir le cadre vu dans la section précédente :

$$P_k = \{x_k + \Delta_k d : d \in D_k\} \tag{2.12}$$

qui constitue un sous-ensemble du treillis. Il est possible alors de donner une vue générique de la méthode GPS dans l'algorithme 3.

Les conditions d'arrêt citées dans la partie précédente pour l'algorithme CS restent valides. L'algorithme GPS est également supporté par des résultats de convergence reposant sur la notion de dérivée directionnelle généralisée [40]. Pour une fonction f Lipschitzienne près d'un point x, la dérivée directionnelle généralisée de f en x dans la direction $d \in \mathbb{R}^n$ est définie par :

$$f^\circ(x; d) = \lim_{y \to x} \sup_{t \searrow 0} \frac{f(y + td) - f(y)}{t} \tag{2.13}$$

Il faut aussi définir la notion de sous-suite raffinante. Pour un sous-ensemble d'indice K, une sous-suite $\{x_k\}_{k \in K}$ est raffinante si $\lim_{k \in K} \Delta_k = 0$. La limite de cette sous-suite raffinante est appelée un point raffiné.

Nous résumons les résultats de convergence [16, 22] pour une fonction f bornée :

- Il existe au moins une sous-suite raffinante $\{x_k\}$ et un point raffiné $\hat{x}$. Si la fonction f est continue alors tous les points raffinés ont la même valeur de l'objectif.
- Si f est Lipschitzienne aux alentours d'un point raffiné $\hat{x}$, alors, pour toute direction d de la base positive D' (une des bases sélectionnées pour une infinité d'itérations), la dérivée directionnelle généralisée satisfait $f^\circ(\hat{x}; d) \geq 0$. Si de plus f est régulière en $\hat{x}$, alors, pour toute direction d de la base positive D', la dérivée directionnelle satisfait $f'(\hat{x}; d) \geq 0$.
- Si f est continûment différentiable, alors, pour tout point raffiné $\hat{x}$, $\nabla f(\hat{x}) = 0$.

Algorithme 3: L'algorithme de recherche généralisée par motifs

1-Initialisations

$k = 0, x_k, \Delta_k, \omega^+$ un entier positif , ω^- un entier négatif

2-Arrêt

Si une condition d'arrêt est satisfaite : Stop. Sinon : aller à **3**

3-Itération k

3.1-Recherche Globale (Optionnelle)

Évaluation de f sur l'ensemble des points de recherche sur M_k (stratégie définie par l'utilisateur)

3.2-Sonde locale (Optionnelle en cas de succès de la recherche globale)

Choix d'un ensemble générateur positif des directions D_k

Évaluation de f sur l'ensemble des points de sonde défini par D_k

Arrêt de la sonde dès qu'un meilleur point est repéré

4-Mises à jour

Si un meilleur point x est trouvé (soit en phase de recherche ou de sonde)

$x_{k+1} \leftarrow x$

Choix de $\omega_k \in \{0, 1, \ldots, \omega^+\}$

Sinon

$x_{k+1} \leftarrow x_k$

Choix de $\omega_k \in \{\omega^-, \omega^- + 1, \ldots, -1\}$

fin Si

$\Delta_{k+1} \leftarrow \tau^{\omega_k} \Delta_k$

$k \leftarrow k + 1$

aller à **2**

Dans la littérature, plusieurs exemples sont cités pour montrer les limites de GPS. L'exemple cité dans [107] met en évidence la faiblesse d'utiliser un nombre fini de directions de recherche et [11] fournit plusieurs exemples qui convergent vers des solutions non optimales.

L'algorithme de recherche par génération d'ensembles (GSS) [107] est une généralisation de GPS, qui permet de remplacer la restriction du treillis par une condition de décroissance minimale sur f. Une méthode de Lagrangien augmenté peut être utilisée pour gérer les contraintes explicites générales [113].

2.3.3 Recherche directe par treillis adaptatifs

MADS est une généralisation de l'algorithme GPS. Il a été développé pour traiter tout type de contraintes et pour pallier les difficultés rencontrées dans les problèmes de boites noires non lisses ou qui présentent des contraintes cachées.

Pour présenter l'algorithme MADS, on redéfinit le treillis :

$$M_k = \{x + \Delta_k^m Dz : x \in V_k, z \in \mathbb{N}^{n_D}\} \tag{2.14}$$

où $\Delta_k^m \in \mathbb{R}^+$ est le paramètre de taille du treillis à l'itération k, V_k est la cache qui contient tous les points évalués au départ de l'itération k et $D \in \mathbb{R}^{n \times n_D}$ la matrice composée des directions de recherche. D est souvent prise comme la matrice $[I_n$ - $I_n]$ où I_n est la matrice identité de taille n. L'évaluation de la boite noire est souvent un aspect très coûteux et la structure de cache permet de de stocker tous les points qui ont été déjà evalués afin d'éviter de les revisiter.

Chaque itération de MADS est composée de trois étapes (voir la figure 2.2) :

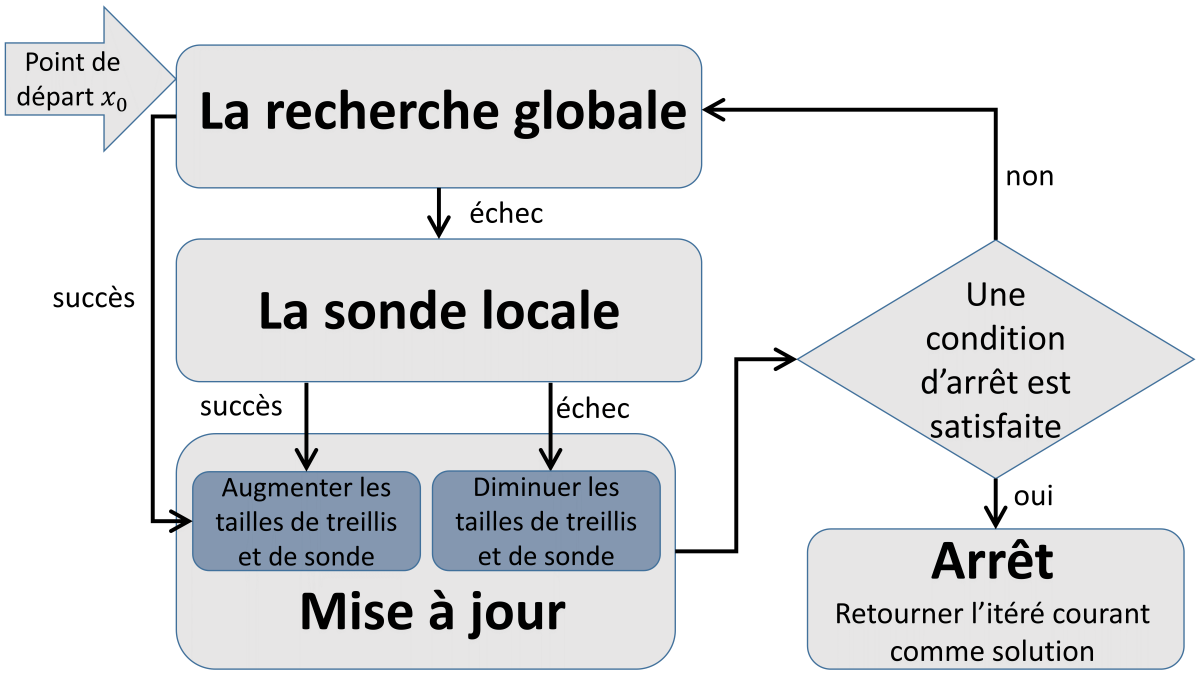


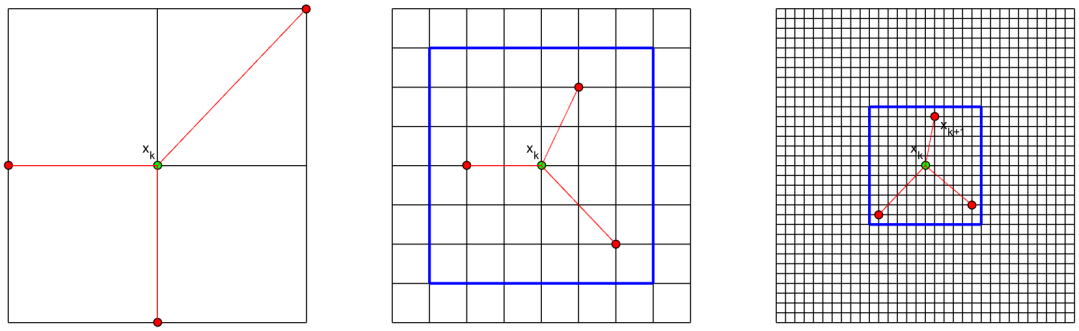
Figure 2.2 L'algorithme de recherche directe par treillis adaptatifs MADS (Figure adaptée de [9]).

- Recherche globale : La recherche est une étape flexible, car elle n'est pas essentielle pour l'analyse de convergence de MADS. Cette étape peut être ignorée, définie spécifiquement pour un problème donné par l'utilisateur, basée sur des fonctions substituts [24, 32] ou des heuristiques [13] pour générer un nombre fini de points à tester sur le treillis. Dans la version actuelle de NOMAD, MADS utilise des modèles quadratiques avec les problèmes de moins de 50 variables pour choisir un candidat prometteur à tester. Si l'étape de recherche réussit, ce

qui veut dire qu'elle trouve une nouvelle meilleure solution, alors l'étape de sonde peut être ignorée et nous passons directement à la phase des mises à jour. Par contre, si la recherche échoue, une sonde locale est initiée.

- Sonde locale : La sonde utilise un sous-ensemble de directions, qui représente une base positive de l'espace des solutions, pour sélectionner plusieurs points d'essai sur le treillis à une distance inférieure à Δ_k^p (en utilisant la norme infinie) de l'itéré courant. Δ_k^p est nommé la taille du cadre de sonde (voir [4, 18]). L'ensemble des directions normalisées utilisées, lors d'une exécution complète de MADS, doit être dense dans la boule unité pour assurer les résultats de convergences les plus forts (voir [18, 19, 21]).

- Mises à jour : Dans cette étape, si la sonde locale échoue à trouver une meilleure solution, alors les paramètres du treillis Δ_k^m et du cadre de sonde Δ_k^p sont réduits pour effectuer une exploration plus proche de la solution courante à la prochaine itération. Par contre, si la recherche globale ou la sonde locale réussissent à trouver un meilleur itéré, alors les paramètres du treillis Δ_k^m et du cadre de sonde Δ_k^p sont augmentés. En s'assurant que la réduction du paramètre de sonde Δ_k^p est moins accrue que la réduction du paramètre de treillis Δ_k^m, le nombre de directions de sonde augmente considérablement.



(a) $\Delta_k^m = 1; \Delta_k^p = 1$ (b) $\Delta_k^m = \frac{1}{4}; \Delta_k^p = \frac{1}{2}$ (c) $\Delta_k^m = \frac{1}{16}; \Delta_k^p = \frac{1}{4}$

Figure 2.3 Exemple de tailles de treillis et de cadres de sonde locale.

La figure 2.3 donne un exemple d'itération de MADS et l'algorithme 4 donne une version simplifiée de MADS.

Les conditions d'arrêt mentionné dans l'algorithme CS restent valide pour MADS. Le plus souvent, l'algorithme MADS s'arrête si un nombre d'évaluations maximal ou si une valeur limite du

Algorithme 4: algorithme MADS

1-Initialisations

$x_0 \in \Omega, \tau \in \mathbb{Q}, l_0 \leftarrow 0, k \leftarrow 0$

2-Arrêt

Si une condition d'arrêt est satisfaite : Stop. Sinon : aller à **3**

3-Itération k

$\Delta_k^m \leftarrow \min\{1, \tau^{-l_k}\}$

$\Delta_k^p \leftarrow \tau^{\frac{-l_k}{2}}$

3.1-Recherche Globale (Optionnelle)

évaluation de f et c sur un ensemble fini $S_k \subset M_k$

3.2-Sonde locale (Optionnelle en cas de succès de la recherche globale)

Calcul des directions de recherche D_k

évaluation de f et c sur un ensemble fini $P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k$

4-Mises à jour

Si $\exists y \in S_k \cup P_k$ tel que $f(y) < f(x_k)$

$x_{k+1} \leftarrow y$

$l_{k+1} \leftarrow l_k - 1$

Sinon

$x_{k+1} \leftarrow x_k$

$l_{k+1} \leftarrow l_k + 1$

fin Si

$k \leftarrow k + 1$

aller à **2**

paramètre de treillis sont atteints. Même si cet algorithme peut utiliser la barrière extrême [18], MADS se base principalement sur la méthode de la barrière progressive [19] pour le traitement des contraintes.

L'algorithme MADS est supporté par plusieurs résultats de convergence décrits dans [18]. Le premier résultat démontré, pour une fonction avec des courbes de niveau bornées, est :

$$\liminf_{k\to\infty} \Delta_k^m = \liminf_{k\to\infty} \Delta_k^p = 0. \tag{2.15}$$

Le résultat suivant montre que, sous certaines conditions, une fonction converge globalement vers un point critique de Clarke qui n'est pas systématiquement un optimum local. Pour introduire le résultat, il faut définir le cône hypertangent [35]. Pour un point $x \in \Omega$, un vecteur v est dit hypertangent à Ω si et seulement s'il existe $\epsilon > 0$ tel que $y + t\omega \in \Omega$ pour tout $y \in \Omega \cap B(x, \epsilon)$, $\omega \in B(v, \epsilon)$ et $0 < t < \epsilon$. L'ensemble des vecteurs v constitue le cône hypertangent noté $T_\Omega^H(x)$.

Nous pouvons énoncer le résultat de convergence suivant : pour une fonction Lipschitzienne f autour d'un point raffiné $\hat{x} \in \Omega$, si $v \in T_\omega^H(\hat{x})$ est une direction raffinante, alors : $f^\circ(\hat{x}; v) \geq 0$.

Dans la suite de cette revue de littérature, nous allons détailler deux extensions de MADS qui sont étudiées dans les chapitres 4 et 6 : l'utilisation des modèles quadratiques dans l'étape de recherche de MADS [49] et décomposition parallèle de l'espace pour MADS (PSD-MADS [20]). Parmi les autres extensions de MADS, on compte l'utilisation de l'approche VNS dans l'étape de recherche pour échapper aux minimums locaux [13], la prise en charge des variables catégoriques [3] (variables discrètes qui prennent un nombre fini de valeurs sans aucune propriété d'ordanancement) et périodiques [25], l'utilisation d'un treillis anisotropique [27] (chaque variable a un paramètre de treillis différent) et le traitement de problèmes d'optimisation multiobjective [28] et robuste [23].

2.4 Modèles quadratiques dans MADS

L'utilisation des modèles quadratiques pour améliorer MADS a été introduite dans [49]. Pour décrire cette approche, nous commençons par considérer la base canonique de l'espace des polynômes de degré inférieur ou égal à 2 dans $\mathbb{R}^n$:

$$\begin{aligned}\phi(x) &= (\phi_0(x), \phi_1(x), \ldots, \phi_q(x))^\top \\ &= \left(1, x_1, x_2, \ldots, x_n, \frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2}, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n\right)^\top.\end{aligned} \tag{2.16}$$

La base contient $q+1 = (n+1)(n+2)/2$ éléments. Pour une fonction f donnée, il est possible de trouver, à l'itération k, un unique modèle quadratique f^k défini avec $q+1$ paramètres $\alpha \in \mathbb{R}^{q+1}$:

$$f^k(x) = \alpha^\top \phi(x). \tag{2.17}$$

Donc, pour obtenir une approximation de la fonction objectif f, il faut définir un ensemble de $p+1$ points d'interpolation $Y = \{y^0, y^1, \ldots, y^p\}$.

MADS se base sur la cache pour sélectionner cet ensemble d'interpolation. Dans l'étape de recherche, l'ensemble d'interpolation est choisi en considérant tous les points de la cache qui sont à l'intérieur de la boule centrée autour de l'itéré courant et de rayon égal au double du paramètre de sonde. Dans la phase de sonde, l'ensemble d'interpolation contient tous les points de cache dans la boule centrée autour de l'itéré courant et de rayon $2r$, avec r le plus petit rayon des boules contenant tous les points à tester.

La construction du modèle se fait en déterminant le vecteur α qui va minimiser $\sum_{y \in Y} (f(y) - f^k(y))^2$. Ceci revient à résoudre, au sens des moindres carrés, le système :

$$M(\phi, Y)\alpha = f(Y) \tag{2.18}$$

avec $f(Y) = (f(y^0), f(y^1), \ldots, f(y^p))^\top$ et

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \ldots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \ldots & \phi_q(y^1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \ldots & \phi_q(y^p) \end{bmatrix} \in \mathbb{R}^{(p+1)\times(q+1)}. \tag{2.19}$$

La résolution de ce problème est détaillée dans les deux sous-parties suivantes. (Pour une analyse similaire dans le cas linéaire, voir [144])

2.4.1 Modèle de régression

Quand $p \geq q$, le système (2.18) est dit surdéterminé puisqu'il y a plus de points d'interpolation que nécessaire (dans le cas où $p = q$ le système est dit déterminé, mais se résout de la même façon que pour $p > q$). Donc, il faut utiliser une régression et résoudre le problème :

$$\min_{\alpha \in \mathbb{R}^{q+1}} \|M(\phi, Y)\alpha - f(Y)\|^2 \tag{2.20}$$

avec $\|.\|$ représentant la norme euclidienne. Dans le cas où $M(\phi, Y)$ est de plein rang, il existe une solution unique au problème de régression (2.20) :

$$\alpha = [M(\phi, Y)^\top M(\phi, Y)]^{-1} M(\phi, Y)^\top f(Y). \tag{2.21}$$

2.4.2 Modèle MFN (*Minimum Frobenius Norm*)

Quand $p < q$, le système (2.18) est dit sous-déterminé puisqu'il n'y a pas assez de points d'interpolation et dans ce cas, il existe une infinité de solutions. Le système va être régularisé, dans ce cas, en minimisant la norme de Frobenius du problème :

$$\begin{aligned} \underset{\alpha_Q \in \mathbb{R}^{n_Q}}{\text{minimize}} \quad & \frac{1}{2}\|\alpha_Q\|^2 \\ \text{sujet à} \quad & M(\phi_L, Y)\alpha_L + M(\phi_Q, Y)\alpha_Q = f(Y) \end{aligned} \tag{2.22}$$

avec $n_Q = n(n+1)/2$, $\alpha = \begin{bmatrix} \alpha_L \\ \alpha_Q \end{bmatrix}$, $\alpha_Q \in \mathbb{R}^{n_Q}$, $\alpha_L \in \mathbb{R}^{n+1}$, $\phi_L = (1, x_1, x_2, \ldots, x_n)^\top$ et $\phi_Q = \left(\frac{x_1^2}{2}, \frac{x_2^2}{2}, \ldots, \frac{x_n^2}{2}, x_1x_2, x_1x_3, \ldots, x_{n-1}x_n\right)^\top$. Le problème (2.22) admet une solution unique si

la matrice :

$$F(\phi, Y) = \begin{bmatrix} M(\phi_Q, Y)M(\phi_Q, Y)^\top & M(\phi_L, Y) \\ M(\phi_L, Y)^\top & 0 \end{bmatrix} \in \mathbb{R}^{(p+n+2)\times(p+n+2)} \tag{2.23}$$

est non singulière. Pour trouver une solution au problème (2.22), il faut d'abord déterminer α_L via le système :

$$F(\phi, Y) \begin{bmatrix} \mu \\ \alpha_L \end{bmatrix} = \begin{bmatrix} f(Y) \\ 0 \end{bmatrix} \tag{2.24}$$

où $\mu \in \mathbb{R}^{p+1}$ sont les multiplicateurs de Lagrange du problème (2.22), ensuite α_Q est donné par la relation suivante :

$$\alpha_Q = M(\phi_Q, Y)^\top \mu. \tag{2.25}$$

Ceci complète la construction du modèle.

2.4.3 Intégration des modèles quadratiques à MADS

Dans la suite de ce chapitre, à chaque itération k, l'étape de recherche construit des modèles quadratiques locaux de la fonction objectif f et des m contraintes d'inégalité autour de l'itéré courant x^k. Les modèles construits forment un problème quadratique avec contraintes quadratiques et une région de confiance en norme ℓ_∞ :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f^k(x) \\ \text{sujet à} & c_j^k(x) \le 0, \qquad j \in \{1, 2, \ldots, m\} \\ & \left\|x - x^k\right\|_\infty \le \delta^k \end{array} \tag{2.26}$$

où le scalaire δ^k définit la région de confiance, f^k est le modèle quadratique de la fonction objectif autour de l'itéré courant x^k et les c_j^k, $j \in \{1, 2, \ldots, m\}$, sont les modèles quadratiques des m contraintes d'inégalité. Nous préférons une région de confiance en norme ℓ_∞ parce qu'il est facile de la représenter par de simples contraintes de bornes. De plus, une telle norme est compatible avec le treillis.

Ces modèles quadratiques sont aussi utilisés dans la sonde locale. Comme nous l'avons vu précédemment, la sonde produit une liste de points à tester sur le treillis, mais, au lieu de les évaluer directement, MADS utilise les modèles quadratiques pour trier les points et envoyer les candidats les plus prometteurs en premier. Cette approche est utilisée avec la stratégie opportuniste pour améliorer les performances de MADS. L'algorithme suivant présente les différences principales par rapport à l'algorithme MADS précédent.

Algorithme 5: Intégration des modèles quadratiques dans MADS (seules les différences par rapport à l'algorithme précédent sont montrées)

1-Initialisations
2-Arrêt
3-Itération k
3.1-Recherche Globale
$S_k \leftarrow \emptyset$
Sélection d'un ou deux centres de modèles
Pour chaque centre :
Construction du centre d'interpolation Y
Construction de $m+1$ modèles (f^k et $c_j^k, j = 1, 2, \ldots, m$)
Minimisation de (4.3)
Sélection d'au plus 2 points tests x_m^f et x_m^i de M_k à ajouter à S_k
3.2-Sonde locale (Optionnelle en cas de succès de la recherche globale)
Calcul de P_k et tri des modèles :
Construction de $m+1$ modèles
Évaluation des modèles sur P_k
Tri des points de P_k
Évaluation de f et c sur P_k
4-Mises à jour

2.5 Résolution des sous-problèmes quadratiques

Les sous-problèmes produits par MADS lors de l'utilisation des modèles quadratiques sont similaires aux sous-problèmes qui surgissent lors de l'utilisation des méthodes de région de confiance [48]. C'est pourquoi, depuis les années quatre-vingt, un grand nombre d'algorithmes ont été développés pour traiter ce genre de problèmes quadratiques. L'algorithme de Moré & Sorenson [129] est l'un des premiers algorithmes à avoir été créé spécifiquement pour les problèmes quadratiques sujets à une contrainte ellipsoïdale. Cette méthode se base sur la résolution des conditions d'optimalité via la méthode de Newton avec backtracking. Par contre, le problème avec cette méthode est l'utilisation de factorisations de Cholesky qui devient un handicap pour de très grandes matrices. Un peu plus tard, l'algorithme de région de confiance généralisée de Lanczos (GLTR) [77] voit le jour en utilisant l'algorithme de Moré & Sorenson sur les sous-espaces de Krylov. Mais, GLTR ne pouvait pas prendre en charge les cas difficiles des problèmes de région de confiance. Le même principe est appliqué par la méthode des sous-espaces séquentiels (SSM) [88] qui utilisent des sous-espaces spécifiques de dimension 4 au lieu de compter sur les sous-espaces de Krylov. Ce changement de sous-espaces permet à SSM de résoudre les problèmes qui présentent des cas difficiles en région de confiance, cependant, SSM ne peut prendre en charge que des régions de confiance sphérique. Récemment, l'algorithme de Gould-Robinson & Thorne [80] a permis de réinventer l'approche

de Moré & Sorenson en utilisant des approximations polynomiales de grandes dimensions permettant ainsi à la méthode de Newton de converger en un nombre réduit d'itérations. Pour les problèmes quadratiques avec une contrainte quadratique convexe, l'algorithme de Fortin-Rendl et Wolkowicz [66, 146] permet de les réécrire sous forme d'un sous problème de valeurs propres qui peut être traité par une méthode de Newton combinée aux conditions d'Armijo-Goldstein. Cet algorithme est propice aux problèmes avec des matrices de grandes tailles.

Tous les algorithmes ci-dessus sont utilisés pour des problèmes avec une fonction objectif quadratique sujet à une seule contrainte quadratique définissant la région de confiance. Dans notre cas, les sous-problèmes sont restreints par m contraintes quadratiques : c'est un sous problème de région de confiance généralisée. Récemment, une méthode a été spécifiquement développée pour ce genre de problèmes par extension de l'algorithme Fortin-Rendl et Wolkowicz [138].

Dans notre cas, nous préférons utiliser des méthodes d'optimisation non linéaire pour traiter le problème avec ou sans contraintes. Ces méthodes permettent d'optimiser d'autres types de sous-problèmes non linéaires dans le cas où l'utilisateur veut utiliser un type de modèle différent. Nous nous intéressons particulièrement à la fonction de pénalité exacte [41, 42, 43] et à la méthode du Lagrangien augmenté [10, 45, 47, 136] que nous détaillerons au chapitre 4.

2.6 Méthodes de décomposition parallèle de l'espace pour l'optimisation non linéaire

L'algorithme de Jacobi par blocs [29] permet de décomposer un problème en plusieurs petits sous-problèmes. Les sous-problèmes peuvent ainsi être résolus de manière parallèle sur plusieurs processeurs. La méthode de décomposition parallèle des variables (PVD) [61] traite des problèmes sans contraintes ou d'instances convexes avec des contraintes séparables par blocs. Elle utilise les mêmes étapes que l'approche de Jacobi par blocs à part que PVD peut modifier légèrement les variables fixées en rajoutant des termes nommés les termes "*forget-me-not*" aux sous-problèmes. Soit I l'ensemble des indices des variables utilisées dans (2.1). Chaque processeur p est responsable de modifier son ensemble de variables primaires (qu'on notera I_p) alors que le reste des variables varie de manière plus restreinte. Cet algorithme consiste en deux phases simples : d'abord une phase de parallélisation où les sous-problèmes sont résolus, suivie par une phase de synchronisation qui permet de choisir le meilleur point parmi les solutions obtenues par chaque processeur. Les termes "*forget-me-not*" rendent l'algorithme plus robuste et plus rapide à converger. L'algorithme PVD peut échouer dans le cas où les contraintes ne peuvent pas être séparées en blocs et dans ce cas Ferris et Mangasarian [61] recommandent l'utilisation d'une méthode duale de pénalité exacte.

Dans [119], l'algorithme de distribution parallèle des gradients (PGD) est introduit pour l'optimisation sans contraintes. Cet algorithme fournit des portions différentes du gradient aux divers

processeurs et chaque processus peut utiliser sa propre direction de recherche et son propre pas sans se soucier des autres processus : c'est pour cela qu'une étape de synchronisation est nécessaire.

Solodov généralise l'algorithme PVD en remplaçant les solutions exactes des sous-problèmes par des conditions de descente suffisantes [157] afin d'améliorer la charge de travail des différents processeurs et de modifier certaines conditions de manipulation des variables secondaires [158] pour permettre de traiter des problèmes avec des contraintes convexes généralisées. Une méthode se basant sur la décomposition de l'espace en parallèle dans [68] combine les avancées de PVD avec les méthodes algébriques de Schwarz.

Fukushima [69] décrit un cadre algorithmique de transformation parallèle des variables (PVT) pour l'optimisation sans contraintes où il montre que PVD non contraint et PGD sont des cas particuliers de PVT. De plus, dans [166], il propose des stratégies pratiques pour l'implémentation des algorithmes de type PVT et effectue des tests de la méthode de Jacobi par blocs (qui est aussi un cas particulier de PVT) sur le superordinateur Fujitsu VPP500.

Trois autres algorithmes sont proposés dans [114] en se basant sur les approches PGD et PVT. Dans [148], deux versions de PVD pour l'optimisation contrainte sont présentées : La première s'attaque aux problèmes avec des contraintes séparables par blocs en utilisant une approche quadratique séquentielle (SQP). La seconde méthode traite les problèmes avec des contraintes convexes générales en utilisant des approximations des directions de gradient projeté. La première méthode de [148] est améliorée dans [89] en utilisant une nouvelle recherche linéaire dans SQP. Dans [87], une méthode basée sur PVD est décrite pour trouver une solution approximative des problèmes des moindres carrés.

2.7 Parallélisme en DFO

La parallélisation en DFO a commencé à la fin des années quatre-vingt et une des premières méthodes a être parallélisée est l'algorithme de Nelder-Mead [135]. Les résultats numériques de cette version parallèle ont montré, dans le cas différentiable, que la méthode Nelder-Mead converge vers un point qui n'est pas un optimum local [160]. Ceci a aussi mené à la conception d'un algorithme de recherche multidirectionnel [56] pour les problèmes sans contraintes qui est supporté par une analyse de convergence.

L'article [71] décrit un algorithme parallèle pour l'optimisation sans contraintes. Cette approche se base sur une condition nécessaire non lisse pour trouver un point avec une valeur de la fonction objectif suffisamment petite et mène à la conception d'une autre méthode DFO pour les problèmes avec contraintes de borne [70].

L'algorithme de recherche par motifs parallèle asynchrone (APPS) est introduit pour la première

fois dans [96] et implémenté dans le logiciel APPSPACK pour l'optimisation de boite noire [82] avec des améliorations telle l'affectation dynamique des évaluations aux processus [106]. Dans [84], une version parallèle de l'algorithme de recherche par génération d'ensembles (GSS) [107] est inclue dans APPSPACK. L'algorithme APPS est intégré dans le logiciel de recherche parallèle par optimisation hybride (HOPSPACK) [137]. Cette structure simplifie l'hybridation de plusieurs méthodes et heuristiques d'optimisation dans un seul algorithme dont l'évaluation de la fonction peut être parallélisée afin d'exploiter la puissance des différentes approches. Une méthode DFO hybride combinant l'algorithme DIRECT [101] et GSS [107] est décrite dans [83].

L'article [143] introduit deux implémentations parallèles de deux méthodes classiques : l'algorithme de Hooke et Jeeves [95] et l'heuristique du recuit simulé [104]. Les deux nouvelles méthodes parallèles ont pour but d'exploiter complètement la puissance de calcul sans dégrader la qualité de la solution. Une méthode de type diviser pour régner est conçue dans [125] pour les boites noires sans contraintes de grande taille. Cette méthode identifie les sous-problèmes indépendants qui sont traités par la méthode des stratégies d'évolution avec adaptation de matrice de covariance (CMA-ES) [90].

Plusieurs versions parallèles des méthodes de descente par coordonnées existent telle la version avec blocs distribués en parallèle [123]. Une autre version est décrite dans [147] et se base sur une loi de probabilité arbitraire pour choisir aléatoirement un ensemble de variables à mettre à jour à chaque itération. L'article [134] définit une structure d'analyse de convergence pour différentes classes parallèles de descente par coordonnées.

Deux versions de la méthode non lisse de recherche par coordonnées sans dérivées (CS-DFN [59]) sont introduites dans [115] et utilisées dans deux approches parallèles hybrides se basant sur une recherche linéaire ainsi que sur le logiciel NOMAD [109]. L'étude menée dans [115] conclut que combiner ces algorithmes donne de meilleurs résultats que de les utiliser de manière autonome.

L'article [14] utilise NOMAD pour étudier la répartition des ressources disponibles afin de décider s'il vaut mieux paralléliser l'algorithme ou la boite noire. Dans [108], une nouvelle méthode multistart en DFO est conçue pour trouver plusieurs minimums locaux pour les problèmes avec contraintes de borne.

2.8 L'algorithme PSD-MADS

PSD-MADS [20] est une version parallèle asynchrone de MADS qui est basée sur l'approche de Jacobi par blocs. PSD-MADS garde la notion d'itération en la définissant comme deux appels successifs du processus maître au sondeur (qui est un processus spécial). Si p est le nombre de processus disponibles, alors PSD-MADS consacre $p-3$ processus pour traiter les sous-problèmes,

un sondeur pour le problème initial, un serveur cache et un serveur maître pour réguler le tout (Voir la figure 2.4).

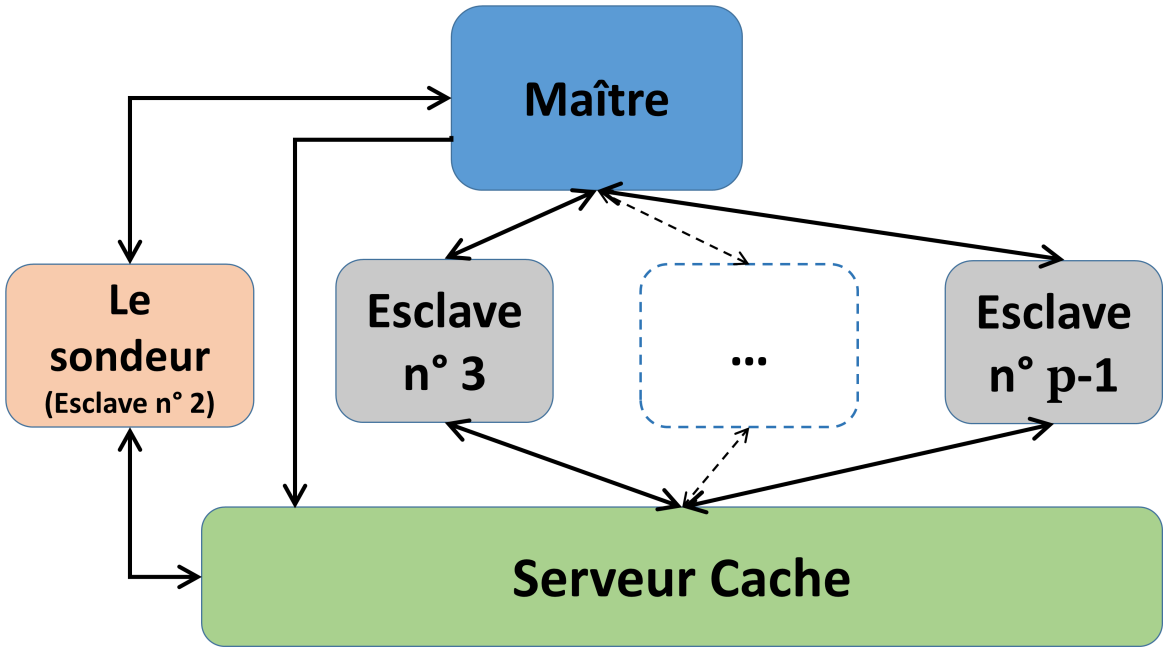


Figure 2.4 Les interactions entre les différents p processus dans PSD-MADS.

Le sondeur est un processus esclave spécial qui optimise le problème initial (2.1) avec toutes les variables : il utilise une seule sonde de l'algorithme MADS qui assure une seule évaluation de la boite noire. Ce processus est important parce qu'il exécute un algorithme MADS valide qui permet de conserver les résultats de convergence de [18].

Les autres processus esclaves optimisent des sous-problèmes définis par une sélection aléatoire des variables de décision en utilisant MADS.

Le serveur cache est un processus spécial qui mémorise toutes les évaluations effectuées par la boite noire : les processus esclaves peuvent ainsi interroger ce processus pour vérifier si un point n'a pas été déjà évalué avant de lancer l'évaluation.

Le processus maître gère tous les processus esclaves : il assigne les sous-problèmes et les paramètres de MADS pour chaque processus, attend les résultats et met à jour les données.

L'utilisateur peut définir deux paramètres dans PSD-MADS : le premier paramètre $\mathcal{P}_{ns}$ est le nombre de variables sélectionnées aléatoirement pour construire un sous-problème et le second paramètre $\mathcal{P}_{bbe}$ est le nombre maximum d'évaluations de boites noires permises pour résoudre cha-

cun des sous-problèmes via MADS. Dans [20], les meilleurs résultats sont obtenus pour $\mathcal{P}_{bbe} = 10$ et $\mathcal{P}_{ns} = 2$. Nous testons quelques valeurs supplémentaires de ces deux paramètres dans la section 6.3.

2.9 Méthodes de sélection de variables

Les méthodes de sélection de variables ont pour but de trouver les variables les plus importantes d'un problème afin de réduire sa dimension. Il est possible de définir deux types de sélection de variables pour des problèmes comportant une ou plusieurs réponses.

Pour des problèmes à une seule réponse, la sélection de variables se fait par des techniques comme l'analyse de variance (ANOVA), l'analyse par composantes principales (PCA) [100] ou l'analyse de sensibilité [34, 132, 155].

Si $Y = [y_1\, y_2\, \ldots\, y_N]$ est une matrice constituée de N observations centrées de dimension n (c'est le cas des entrées de la cache de PSD-MADS pour un problème avec des variables mise à l'échelle), PCA [100] commence par évaluer la matrice de covariance $cov(Y)$ et effectuer une analyse de valeurs et vecteurs propres :

$$cov(Y) \;=\; \frac{1}{N-1} YY^\top \;=\; QDQ^\top \tag{2.27}$$

où D est la matrice diagonale contenant les valeurs propres de $cov(Y)$ dans un ordre descendant et $Q = [v_1\, v_2\, \ldots\, v_n]$ est constituée des vecteurs propres orthogonaux correspondants. En sélectionnant les $p < n$ premiers vecteurs (correspondant aux k plus grandes valeurs propres), la matrice $Q_p = [v_1\, v_2\, \ldots\, v_p]$ est utile pour évaluer les p composantes principales $P = Q_p^\top Y$ qui transforment les données de départ de dimension n en un ensemble de données de taille $p < n$. L'analyse des éléments principaux (PFA) [116] utilise PCA pour évaluer la matrice $Q_p^\top$ précédente puis fait appel à k-mean pour classer les lignes de cette matrice en plusieurs groupes. La ligne la plus proche du centroïde de chaque groupe est choisie et la variable correspondante à cette ligne est considérée comme une variable importante. Donc, en utilisant k groupes, il est possible de choisir k variables importantes. Cette méthode est utilisée dans [116] pour sélectionner des points caractéristiques de mouvement facial et dans [164] pour analyser les images par résonance magnétique fonctionnelle pour le décodage cérébral.

L'analyse de sensibilité se divise en deux catégories : des méthodes locales et globales. Les méthodes locales vérifient l'effet d'une variable dans le voisinage d'un point donné. Cette catégorie inclut l'analyse par différences finies, l'utilisation de diagrammes tornades [97] (diagramme à bandes verticales qui permet de comparer l'importance des variables) et les méthodes basées sur la diffé-

rentiation [92]. En revanche, les méthodes globales prennent en compte la variation d'un élément sur l'espace en entier. Quelques-unes de ces approches sont basées sur l'analyse de la variance telles que la méthode Sobol [156] ou la méthode FAST (*Fourier Amplitude Sensitivity Test*) [53]. D'autres méthodes sont basées sur les fonctions substituts [142], sur la régression multilinéaire [149] ou bien sur la densité [33].

Pour les problèmes à réponses multiples, plusieurs stratégies existent pour sélectionner les variables. Par exemple, il est possible de sélectionner les variables les plus importantes pour chaque réponse ou bien d'éliminer celles-ci et ne garder que les variables non importantes. Il est aussi possible de faire une sélection des variables suivant leurs effets moyens sur toutes les réponses du problème. Pour plus de détail sur les méthodes de sélection de variables dans les problèmes multi réponses, voir [34].

STATS-MADS [5] est une version de MADS pour des problèmes d'optimisation de grande taille sans contraintes. Cette méthode se base sur une étude statistique pour identifier les variables dont l'effet sur la fonction objectif est important et les organiser dans des sous-espaces. Cette approche alterne l'optimisation sur l'espace des solutions en entier (pour rassembler des informations sur l'influence des variables) et l'optimisation sur les sous-espaces (étape d'intensification pour explorer des régions prometteuses). L'implémentation de STATS-MADS utilise une méthode d'analyse de sensibilité pour la sélection de variables. STATS-MADS est testé dans [5] sur des problèmes de dimension allant jusqu'à 500 variables. Cet article conclut en mentionnant l'intérêt d'utiliser une telle technique dans un contexte de parallélisme.

Les méthodes de sélection de variables et les méthodes de décomposition en sous-espaces sont citées parmi les cinq stratégies principales pour s'attaquer aux problèmes de grande taille [152]. Les méthodes de décomposition en sous-espaces semblent les plus prometteuses vue qu'elles permettent d'utiliser un environnement parallèle pour optimiser des sous-problèmes de plus petite taille via différentes méthodes et de manière concurrente. Malgré ces avantages, l'article [152] soulève la question de la manière de décomposition d'un problème en prenant en compte la corrélation entre ses variables et ses réponses. L'approche détaillée dans cette section se base sur la décomposition parallèle de l'espace décrite dans [20] et combine une méthode de sélection de variables adaptée de [5] avec une technique de classification non supervisée pour guider la décomposition.

CHAPITRE 3 ORGANISATION DE LA THÈSE

L'objectif principal de cette thèse est d'améliorer des extensions de MADS pour résoudre des problèmes plus efficacement et s'attaquer à des instances de tailles plus grandes que celles traitées habituellement par les méthodes DFO ou BBO. La première extension (chapitre 4) se rapporte à celles des modèles quadratiques lors de l'étape de recherche de MADS alors que la seconde (chapitre 6) concerne l'algorithme PSD-MADS et la manière dont nous sélectionnons les variables pour construire les sous-problèmes à optimiser parallèlement. Les travaux de recherche réalisés pour la première extension mènent à l'établissement d'une nouvelle méthode d'optimisation non linéaire qui est développée plus en détail dans le chapitre 5.

Le chapitre 4 vise à trouver une meilleure méthode pour résoudre les sous-problèmes quadratiques qui surgissent lors de l'utilisation de MADS avec des modèles. Deux méthodes d'optimisation non linéaire classiques et une nouvelle approche sont décrites et intégrées dans NOMAD pour traiter les sous-problèmes de MADS. Les deux méthodes classiques sont les algorithmes de la fonction de pénalité exacte en norme ℓ_1 et du Lagrangien augmenté. En combinant les points forts des deux méthodes précédentes, la nouvelle méthode du Lagrangien augmenté en norme ℓ_1 se base sur l'approche de Coleman et Conn [42, 43] comme boucle interne de l'algorithme (c'est la boucle interne de la fonction de pénalité exacte en norme ℓ_1) et sur une itération externe semblable à celle du Lagrangien augmenté standard. Le chapitre conclut avec une comparaison numérique entre ces trois méthodes et l'utilisation de MADS ou IPOPT [163] pour la résolution des sous-problèmes quadratiques dans MADS.

Le chapitre 5 reprend l'algorithme du Lagrangien augmentée en norme ℓ_1 décrit dans le contexte DFO au chapitre 4 et développe une analyse de convergence pour les boucles interne et externe de la méthode. Pour la boucle interne, basée sur l'approche de Coleman et Conn, les résultats de convergence montrent que la méthode génère une suite de points qui convergent vers un point stationnaire et que cette convergence est superlinéaire en deux étapes. Pour la boucle externe, l'algorithme du Lagrangien augmenté est décrit pour des contraintes d'égalité avec des contraintes de borne simplifiées ($\mathcal{X} = \{x \in \mathbb{R}^n : x \geq 0\}$). Les résultats de convergence démontrés prouvent que le paramètre de pénalité est borné loin de zéro, les estimations des multiplicateurs convergent vers les vraies valeurs des multiplicateurs de Lagrange et la suite d'itérés générée par la méthode converge vers un point de Karush-Kuhn-Tucker. De plus, la vitesse de convergence de ces suites est R-linéaire.

Le chapitre 6 se concentre sur l'amélioration de la sélection de variables pour la construction de sous-problèmes dans PSD-MADS. La méthode commence par construire une matrice nommée matrice de sensibilité inspirée des travaux sur STATS-MADS [5]. Cette matrice condense l'in-

formation sur l'influence des variables sur la fonction objectif et les contraintes du problème. Le chapitre 6 présente diverses stratégies pour exploiter la matrice de sensibilité et utilise l'algorithme de classification k-mean pour produire des groupes de variables qui partagent plus ou moins le même niveau d'influence sur les sorties du problème. Les résultats numériques sont conduits sur un superordinateur d'Hydro-Québec avec 500 coeurs pour des problèmes allant jusqu'à $4{,}000$ variables.

CHAPITRE 4 TRAITEMENT DES MODÈLES QUADRATIQUES DANS MADS

Les travaux décrits dans ce chapitre ont été publiés dans la revue *European Journal of Operational Research* [9]. On considère le problème d'optimisation avec contraintes suivant :

$$\min_{x\in\Omega} \quad f(x), \tag{4.1}$$

avec :

$$\Omega = \{x \in \mathcal{X} : c_j(x) \leq 0, j \in \{1, 2, \ldots, m\}\}, \tag{4.2}$$

où m est un entier positif, $\mathcal{X}$ est un sous-espace de $\mathbb{R}^n$ qui est souvent défini par des bornes sur les variables, f et c_j, $j \in \{1, 2, \ldots, m\}$, sont des fonctions réelles évaluées par des simulations informatiques et considérées comme des boites noires.

Ce chapitre se concentre sur l'algorithme de recherche directe sur treillis adaptatifs (MADS) [18] avec des modèles quadratiques [49]. MADS se base essentiellement sur deux étapes, nommées la recherche et la sonde, pour explorer l'espace des solutions et une troisième étape pour faire la mise à jour des paramètres de la méthode. La recherche et la sonde sont deux étapes complémentaires : l'étape de recherche permet une exploration locale et globale de l'espace de solutions. Alors que la sonde est locale et permet d'assurer la convergence de la méthode, l'étape de recherche n'a pas d'impact sur l'analyse de convergence de MADS mais améliore les résultats de l'algorithme considérablement.

Dans [49], la phase de recherche globale consiste, à chaque itération k, à créer un sous-problème quadratique avec contraintes quadratiques :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f^k(x) \\ \text{sujet à} & c_j^k(x) \leq 0, \qquad\qquad j \in \{1, 2, \ldots, m\} \\ & \left\|x - x^k\right\|_\infty \leq \delta^k \end{array} \tag{4.3}$$

où le scalaire δ^k définit la région de confiance, f^k est le modèle quadratique de la fonction objectif autour de l'itéré courant x^k et les c_j^k, $j \in \{1, 2, \ldots, m\}$, sont les modèles quadratiques des m contraintes d'inégalité.

Avant ce travail, la résolution de ce sous-problème était assurée par l'intermédiaire d'une nouvelle instance simplifiée MADS. La structure du problème n'est pas exploitée étant donné que MADS

n'utilise pas les informations du gradient. Le but principal de ce chapitre est de trouver un algorithme plus approprié pour traiter les sous-problèmes quadratiques dans l'étape de recherche de MADS. Nous choisissons de tester deux méthodes classiques de l'optimisation non linéaire : la fonction de pénalité exacte en norme ℓ_1 et l'algorithme du Lagrangien augmenté. De plus, nous introduisons une nouvelle méthode appelée le Lagrangien augmenté en norme ℓ_1 qui combine les points forts des deux méthodes précédentes. La nouvelle méthode utilise un terme de pénalité en norme ℓ_1 à la place du terme carré présent dans la méthode du Lagrangien augmenté classique. On va explorer cette méthode en détail hors du contexte DFO dans le chapitre 5.

La section 4.1 présente la fonction de pénalité exacte en norme ℓ_1 et la section 4.2 décrit l'algorithme du Lagrangien augmenté. Nous introduisons la nouvelle méthode d'optimisation non linéaire dans la section 4.3 et nous comparons toutes ces approches dans la section 4.4 sur un ensemble de 65 problèmes.

4.1 Fonction de pénalité exacte en norme ℓ_1 (ℓ_1EPF)

La fonction de pénalité exacte en norme ℓ_1 transforme le problème (4.3) en un problème avec contraintes de bornes :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f^k(x) + \frac{1}{\mu}\sum\limits_{j=1}^{m}\max\{0, c_j^k(x)\} \\ \text{sujet à} & \left\|x - x^k\right\|_\infty \leq \delta^k \end{array} \tag{4.4}$$

où $\mu \in \mathbb{R}^+$ est le coefficient de pénalité. Une méthode analogue est discutée dans [2] en 1955. Cependant, c'est Zangwill qui introduit la fonction de pénalité exacte en norme ℓ_1 dans [168, 169] et Pietrzykowski montre dans [159] que, si la fonction objectif et les contraintes sont continûment différentiables, pour μ suffisamment petit, tout minimum local du problème non linéaire (4.3) est un minimum local de la fonction de pénalité exacte en norme ℓ_1 du problème. Quand le problème est convexe [117, 169], il existe une valeur seuil μ_0 telle que, pour $\mu < \mu_0$, un minimum de la fonction de pénalité exacte en norme ℓ_1 est aussi un minimum du problème (4.3). Charalambous généralise ces résultats dans [39] sous des conditions de second ordre. Dans [41], Coleman et Conn exposent des conditions d'optimalité supplémentaires de la fonction de pénalité exacte en norme ℓ_1. Dans ce chapitre, nous implémentons la version de l'algorithme de fonction de pénalité exacte en norme ℓ_1 décrite dans [42, 43].

La fonction objectif de (4.4) est non différentiable sur la frontière du domaine défini par les contraintes $c_j^k \leq 0$, $j \in \{1, 2, \ldots, m\}$. Nous introduisons deux sous-espaces d'indices à un point donné $\tilde{x} \in \mathcal{X}$:

$$A_\epsilon = \left\{ j \in \{1, 2, \ldots, m\} : |c_j^k(\tilde{x})| \leq \epsilon \right\} \quad \text{et} \quad V_\epsilon = \left\{ j \in \{1, 2, \ldots, m\} : \ c_j^k(\tilde{x}) \ > \epsilon \right\}. \quad (4.5)$$

L'ensemble A_ϵ contient les indices des contraintes ϵ-actives en $\tilde{x}$ et V_ϵ groupe les indices des contraintes ϵ-violées. Le reste des contraintes ne contribue pas à la fonction objectif à cause du terme ℓ_1. Donc, dans un voisinage $\mathcal{N}(\tilde{x}) \subseteq \mathcal{X}$ de $\tilde{x}$, le problème (4.4) devient :

$$\begin{array}{ll} \min\limits_{x \in \mathcal{N}(\tilde{x})} & p^k(x, \mu) + \frac{1}{\mu} \sum\limits_{j \in A_\epsilon} \max\{0, c_j^k(x)\} \\ \text{sujet à} & \left\| x - x^k \right\|_\infty \leq \delta^k \end{array} \quad (4.6)$$

où la fonction de pénalité $p^k(x, \mu) := f^k(x) + \frac{1}{\mu} \sum\limits_{j \in V_\epsilon} \max\{0, c_j^k(x)\}$ est différentiable pour tout $x \in \mathcal{N}(\tilde{x})$. La seconde partie de la fonction objectif du problème (4.6) n'est pas localement différentiable. Pour surmonter cette difficulté, il faut chercher une direction de descente $h \in \mathbb{R}^n$ qui empêche tout changement de valeur des contraintes ϵ-actives pendant la minimisation de la variation de la fonction de pénalité :

$$\begin{array}{ll} \min\limits_{h \in \mathbb{R}^n} & \nabla_x p^k(\tilde{x}, \mu)^\top h + \frac{1}{2} h^\top \nabla_{xx} p^k(\tilde{x}, \mu) h \\ \text{sujet à} & \nabla_x c_j^k(\tilde{x})^\top h + \frac{1}{2} h^\top \nabla_{xx} c_j^k(\tilde{x}) h = 0, \quad j \in A_\epsilon. \end{array} \quad (4.7)$$

Pour trouver la solution du problème (4.7), il est possible de faire une approximation (voir [43]) via :

$$\min_{h \in \mathbb{R}^n} \max_{\lambda \in \mathbb{R}^t} \ h^\top (\nabla_x p^k(\tilde{x}, \mu) - \sum_{j \in A_\epsilon} \lambda_j \nabla_x c_j^k(\tilde{x})) + \frac{1}{2} h^\top (\nabla_{xx} p^k(\tilde{x}, \mu) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla_{xx} c_j^k(\tilde{x})) h, \quad (4.8)$$

où t est la taille de l'ensemble ϵ-active et $\bar{\lambda} \in \mathbb{R}^t$ une approximation du vecteur des multiplicateurs de Lagrange pour les contraintes ϵ-actives. Nous discutons la manière d'évaluer $\bar{\lambda}$ un peu plus loin dans cette section. Pour résoudre (4.8), il faut d'abord définir la matrice des gradients des contraintes ϵ-actives $J_{A_\epsilon}^k(x) = [\nabla c_j^k(x)]_{j \in A_\epsilon} \in \mathbb{R}^{n \times (n-t)}$ et, ensuite, trouver une matrice $Z \in \mathbb{R}^{n \times (n-t)}$ telle que :

$$\left\{ \begin{array}{rcl} J_{A_\epsilon}^{k\top} Z & = & 0, \\ Z^\top Z & = & I, \end{array} \right. \quad (4.9)$$

avec I la matrice identité de taille $n - t$. Évaluer la matrice Z se base sur une décomposition QR de la matrice $J_{A_\epsilon}^k$: il existe une matrice orthogonale $Q \in \mathbb{R}^{n \times n}$ et une matrice triangulaire supérieure

$U \in \mathbb{R}^{t\times t}$ telles que :

$$J^k_{A_\epsilon} = QR = [Q_1\ Q_2] \begin{bmatrix} U \\ 0 \end{bmatrix} = Q_1 U. \tag{4.10}$$

La matrice $Z = Q_2$ est une solution du système (4.9). En utilisant le changement de variable $h \leftarrow Zd$, le problème (4.8) devient :

$$\min_{d\in\mathbb{R}^n} \max_{\lambda\in\mathbb{R}^t} \quad d^\top Z^\top(\nabla_x p^k(\tilde{x},\mu) - \sum_{j\in A_\epsilon} \lambda_j \nabla_x c^k_j(\tilde{x})) + \tfrac{1}{2} d^\top Z^\top(\nabla_{xx} p^k(\tilde{x},\mu) - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla_{xx} c^k_j(\tilde{x}))Zd. \tag{4.11}$$

Mais comme $J^{k\top}_{A_\epsilon} Z = 0$, l'équation (4.11) se simplifie en :

$$\min_{d\in\mathbb{R}^n} \quad d^\top Z^\top \nabla_x p^k(\tilde{x},\mu) + \tfrac{1}{2} d^\top Z^\top(\nabla_{xx} p^k(\tilde{x},\mu) - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla_{xx} c^k_j(\tilde{x}))Zd. \tag{4.12}$$

Si $Z^\top(\nabla_{xx} p^k(\tilde{x},\mu) - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla^2_{xx} c^k_j(\tilde{x}))Z \in \mathbb{R}^{(n-t)\times(n-t)}$ est définie positive, la solution de (4.12) s'obtient par :

$$Z^\top(\nabla_{xx} p^k(\tilde{x},\mu) - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla_{xx} c^k_j(\tilde{x}))Zd = -Z^\top \nabla_x p^k(\tilde{x},\mu). \tag{4.13}$$

La direction de descente est alors donnée par :

$$h = Zd = -Z(Z^\top(\nabla_{xx} p^k(\tilde{x},\mu) - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla_{xx} c^k_j(\tilde{x}))Z)^{-1} Z^\top \nabla_x p^k(\tilde{x},\mu). \tag{4.14}$$

Cette direction de descente h permet de réduire la valeur de la fonction de pénalité p^k sans perturber les contraintes ϵ-actives. Quand $\tilde{x} + h$ est proche d'un point satisfaisant les conditions suffisantes de second ordre, une direction v est nécessaire pour améliorer l'activité des contraintes ϵ-actives :

$$\phi^k_{A_\epsilon}(\tilde{x} + h + v) = 0, \tag{4.15}$$

avec $\phi^k_{A_\epsilon}(x) = [c^k_j(x)]_{j\in A_\epsilon}$ le vecteur des contraintes ϵ-actives au point x. Grâce au développement de Taylor au premier ordre, l'équation (4.15) devient :

$$\phi^k_{A_\epsilon}(\tilde{x} + h) + J^k_{A_\epsilon}(\tilde{x})^\top v \approx 0. \tag{4.16}$$

La direction v est obtenue par la résolution du système suivant :

$$J^k_{A_\epsilon}(\tilde{x})^\top v = -\phi^k_{A_\epsilon}(\tilde{x}+h). \tag{4.17}$$

En utilisant la décomposition QR précédente :

$$U^\top Q_1^\top v \;\; = \;\; -\phi^k_{A_\epsilon}(\tilde{x}+h). \tag{4.18}$$

Pour évaluer v, il faut d'abord résoudre le système triangulaire inférieure : $U^\top w = -\phi^k_{A_\epsilon}(\tilde{x}+h)$ avant d'accéder à la solution $v = Q_1^\top w$. L'itéré est alors mis à jour selon la direction $h+v$ avec un pas égal à 1 si la valeur de la fonction de pénalité décroît suffisamment. Dans le cas contraire, une recherche linéaire le long de h est entreprise. L'algorithme 6 donne une vue sommaire de la boucle interne de ℓ_1EPF.

Algorithme 6: Algorithme de fonction de pénalité exacte en norme ℓ_1 (itération interne) [43]

Initialiser : x, $\epsilon \leftarrow 1$, A_ϵ, V_ϵ, $\tau \leftarrow 10^{-5}$
Si x est proche d'un point satisfaisant les conditions de 1^{er} ordre et $\left\|\phi^k_{A_\epsilon}(x)\right\| \leq \tau$
    Arrêt
fin Si
Évaluer h (en résolvant approximativement (4.7))
Si $x+h$ est proche d'un point satisfaisant les conditions de 1^{er} ordre
    Évaluer v (en résolvant (4.17))
    Si $x+h+v$ décroît suffisamment la valeur de p
        $x \leftarrow x+h+v$
    Sinon
        $\epsilon \leftarrow \frac{\epsilon}{2}$
        Mettre à jour A_ϵ et V_ϵ
        Évaluer h (en résolvant approximativement (4.7))
        Calculer le pas β le long de h (algorithme de recherche linéaire)
        $x \leftarrow x+\beta h$
    fin Si
Sinon
    Calculer le pas β le long de h (algorithme de recherche linéaire)
    $x \leftarrow x+\beta h$
fin Si
Aller à **2**

La méthode complète se base sur l'estimation des multiplicateurs de Lagrange pour décider si l'algorithme doit évaluer une direction v ou pas. Ces estimations permettent d'inclure l'information de la courbure des contraintes pour approcher la matrice Hessienne de la fonction de pénalité. Quand l'itéré courant x est proche d'un point critique $\tilde{x}$, il existe un vecteur $\bar{\lambda}$ tel que :

$$\nabla f^k(x) + \frac{1}{\mu} \sum_{j \in V_\epsilon} \nabla c_j^k(x) = \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x). \tag{4.19}$$

En passant à une notation matricielle, l'équation (4.19) devient :

$$J_{A_\epsilon}^k(\tilde{x})\bar{\lambda} = \nabla p^k. \tag{4.20}$$

Encore une fois, via la factorisation QR, le système (4.20) se transforme en :

$$U\bar{\lambda} = Q_1^\top \nabla p^k. \tag{4.21}$$

Il est facile de trouver $\bar{\lambda}$ par la suite, vu que U est triangulaire supérieure. Pour résumer la méthode, si l'itéré est loin d'un point stationnaire, une recherche linéaire suivant la direction de descente $h = -Z(Z^\top \nabla_{xx} p^k(\tilde{x}, \mu)Z)^{-1} Z^\top \nabla_x p^k(\tilde{x}, \mu)$ est effectuée. Sinon, si l'algorithme est proche d'un point stationnaire, les approximations des multiplicateurs de Lagrange $(\bar{\lambda}_j)_{j \in A_\epsilon}$ sont évaluées. Si tous les $(\bar{\lambda}_j)_{j \in A_\epsilon}$ sont dans l'intervalle $\left[0, \frac{1}{\mu}\right]$ alors les directions de descente v et $h = -Z(Z^\top(\nabla_{xx} p^k(\tilde{x}, \mu) - \sum_{j \in A_\epsilon} \bar{\lambda}_j \nabla_{xx} c_j^k(\tilde{x}))Z)^{-1} Z^\top \nabla_x p^k(\tilde{x}, \mu)$ sont calculées. Dans le cas où $h + v$ n'apporte pas une décroissance suffisante à la fonction de pénalité, l'algorithme entreprend une recherche linéaire suivant h. Par contre, s'il existe un indice $j_0 \in A_\epsilon$ tel que $\bar{\lambda}_{j_0} \notin \left[0, \frac{1}{\mu}\right]$, la direction de descente suivante est calculée :

$$h_{j_0} = \sigma_{j_0} Z_{j_0} Z_{j_0}^\top \nabla c_{j_0}^k(x), \tag{4.22}$$

où $\sigma_{j_0} = -sgn(\lambda_{j_0})$, et $Z_{j_0} \in \mathbb{R}^{n \times n-t-1}$ et une sous matrice de Z en enlevant la colonne relative à la contrainte j_0. Cette étape est nommée l'abandon d'une contrainte. La figure 4.1 donne une vue graphique de cette approche qu'on nomme la méthode de Coleman et Conn.

L'algorithme de recherche linéaire utilisé dans cette méthode est conçu pour exploiter le caractère quadratique par morceaux de la fonction de pénalité. L'algorithme 7 met à jour l'itéré courant avec une recherche linéaire le long de la direction h.

Finalement, la partie externe de l'algorithme essaye, à chaque itération, de trouver un minimum de la fonction de pénalité pour une valeur fixe de μ. Ensuite, l'algorithme réduit le paramètre μ si le minimum trouvé n'est pas réalisable pour le problème (4.3) afin de mettre plus de poids sur les contraintes. L'algorithme 8 décrit la boucle externe de la méthode ℓ_1EPF.

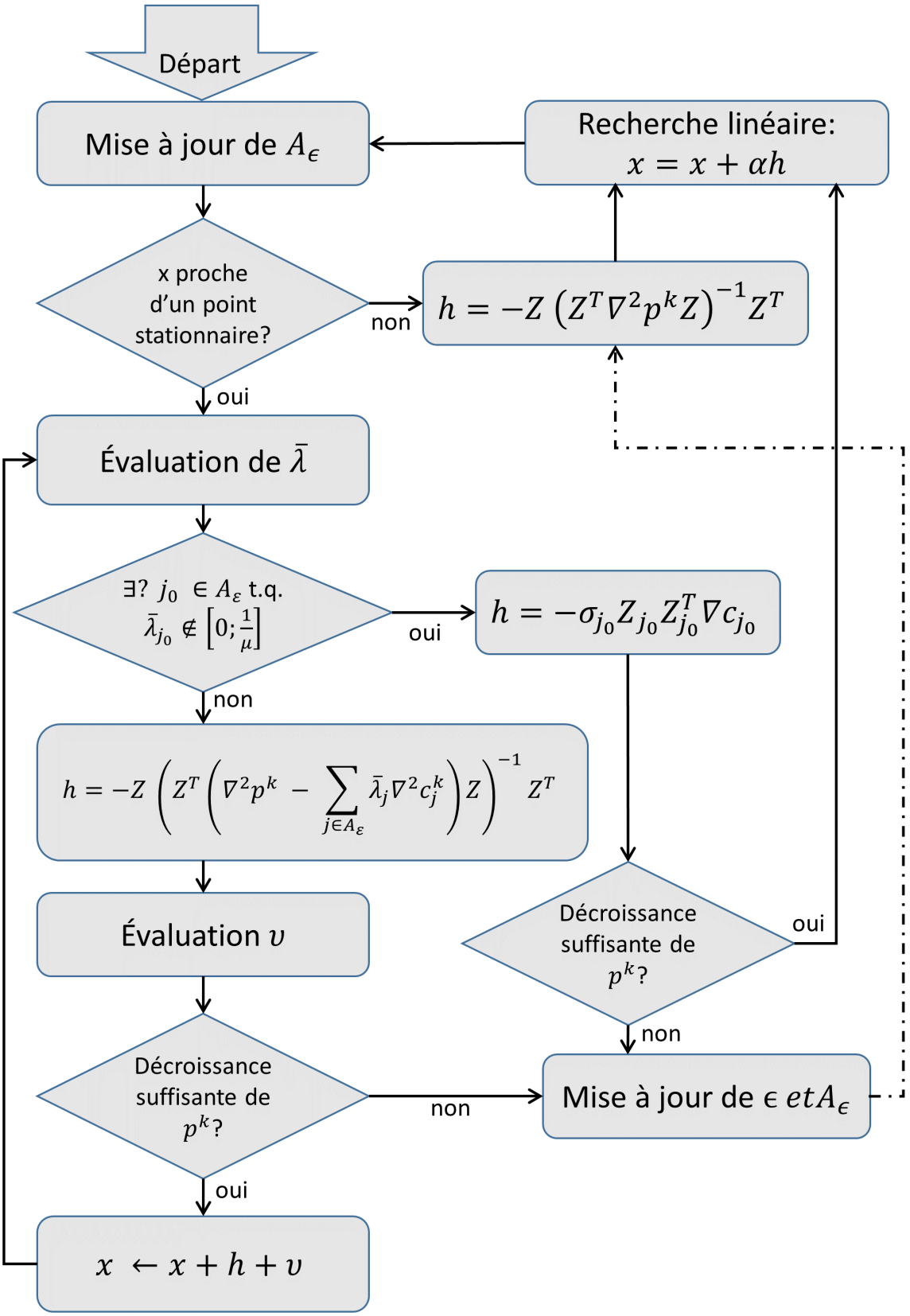


Figure 4.1 La boucle interne de l'algorithme ℓ_1EPF (adaptée de [43]).

Algorithme 7: Algorithme de recherche linéaire pour ℓ_1EPF [43]

$j_{opt} \leftarrow 0$
$\gamma_{j_{opt}} \leftarrow 0$
$a \leftarrow h^\top \nabla p^k$
$I_\epsilon \leftarrow \left\{ j \in \{1, 2, \ldots, m\} : \left| c_j^k(x) \right| > \epsilon \right\}$
$\gamma_j \leftarrow \frac{-c_j^k(x)}{h^\top \nabla c_j^k(x)}$ pour $j \in I_\epsilon$
$B \leftarrow \{j \in I_\epsilon : \gamma_j > 0\}$
Tant que $B \neq \emptyset$ et $a < 0$
    $j_{opt} \in \arg\min_{j \in B} \{\gamma : \gamma \leq \gamma_j\}$
    $a \leftarrow a + \left| h^\top \nabla c_{j_{opt}}^k \right|$
    $B \leftarrow B \setminus \{j_{opt}\}$
fin Tant que
$y \leftarrow x + \gamma_{j_{opt}} h$
Si La fonction de pénalité décroît suffisamment
    $x \leftarrow y$
Sinon
    Lancer une recherche linéaire par ajustement quadratique entre x et $x + \tau h$ avec $\tau > 0$
fin Si

Algorithme 8: Algorithme de fonction de pénalité exacte en norme ℓ_1 (boucle externe) [43]

Initialiser : x, $\mu \leftarrow 1$
Tant que x n'est pas un point réalisable
    Minimiser la fonction de pénalité pour trouver un nouvel itéré x (Algorithme 6)
    Mettre à jour : $\mu = \mu/10$
fin Tant que

4.2 Lagrangien augmenté (AUGLAG)

La méthode du Lagrangien augmenté a été introduite par Powell [139] et Hestenes [93] dans les années soixante. Depuis, la méthode a été intensivement étudiée, améliorée et soutenue par des résultats de convergence [85, 136]. Différentes versions de l'algorithme existent, dont la plus connue est probablement le logiciel LANCELOT [45, 47]. L'utilisation des variables d'écart est populaire avec l'algorithme du Lagrangien augmenté, car l'exploitation de la structure de ces variables permet d'augmenter l'efficacité de la méthode [46, 52]. En DFO, l'article [81] introduit une méthode de Lagrangien augmenté pour les boites noires avec contraintes et [26] conçoit un algorithme de région de confiance sans dérivées (DFTR) qui se base sur un Lagrangien augmenté pour résoudre les sous-problèmes quadratiques avec contraintes quadratiques et mettre à jour le rayon de la région de confiance. Dans cette section, nous nous basons sur l'approche décrite dans [10] qui est similaire à l'implémentation LANCELOT.

Nous rajoutons des variables d'écart au problème (4.3) :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X},\ s\in\mathbb{R}^m} & f^k(x) \\ \text{sujet à} & c_j^k(x) + s_j = 0, \quad j \in \{1,2,\ldots,m\} \\ & \left\|x - x^k\right\|_\infty \leq \delta^k, \\ & s_j \geq 0, \quad\quad\quad j \in \{1,2,\ldots,m\} \end{array} \tag{4.23}$$

Le Lagrangien augmenté transforme le problème (4.3) en :

$$\begin{array}{lrcl} \min\limits_{x\in\mathcal{X},\ s\in\mathbb{R}^m} & \mathcal{L}_a^k(x,s,\lambda,\mu) & = & f^k(x) - \sum\limits_{j=1}^m \lambda_j(c_j^k(x)+s_j) + \frac{1}{2\mu}\sum\limits_{j=1}^m (c_j^k(x)+s_j)^2 \\ \text{sujet à} & \left\|x - x^k\right\|_\infty & \leq & \delta^k, \\ & s_j \geq 0, & & j \in \{1,2,\ldots,m\} \end{array} \tag{4.24}$$

où $\lambda \in \mathbb{R}^m$ et $\mu \in \mathbb{R}^+$. À chaque itération l, l'algorithme du Lagrangien augmenté résout le problème (4.24) pour des valeurs fixées de λ^l et μ^l afin d'évaluer le prochain itéré (x^{l+1}, s^{l+1}). Il faut rappeler que l'indice k caractérise le sous-problème (4.3) alors que l'indice l représente les itérations externes de l'algorithme AUGLAG. La partie externe de l'algorithme met à jour les paramètres λ^l or μ^l : Quand les valeurs des contraintes sont en dessous d'une tolérance donnée, η^l, l'itéré est considéré localement réalisable et l'algorithme met à jour les multiplicateurs λ^l. Dans l'autre cas, la réduction du paramètre de pénalité μ permet de mettre plus de poids sur les contraintes pour essayer de les satisfaire à la prochaine itération. L'algorithme 9 donne une vue générale de l'itération externe de la méthode AUGLAG avec $\mathcal{L}^k(x,s,\lambda) = f^k(x) - \sum\limits_{j=1}^m \lambda_j(c_j^k(x)+s_j)$ et $\mathcal{P}_{\mathcal{X}\times\mathbb{R}_+^m}(y)$ est la projection de y sur l'ensemble $\mathcal{X}\times\mathbb{R}_+^m = \{(x,s) : x \in \mathcal{X}, s \in \mathbb{R}_+^m\}$.

Pour évaluer l'itéré suivant dans la ligne 7 de l'algorithme 9, à chaque itération l de l'itération k externe, un modèle quadratique est utilisé comme approximation du problème (4.24) :

$$\begin{array}{ll} \min\limits_{d^x\in\mathbb{R}^n,\ d^s\in\mathbb{R}^m} & \nabla_x\mathcal{L}_a^k(x^l,s^l,\lambda^l,\mu^l)^\top d^x + \frac{1}{2}d^{x\top}\nabla_{xx}\mathcal{L}_a^k(x^l,s^l,\lambda^l,\mu^l)d^x + \\ & \nabla_s\mathcal{L}_a^k(x^l,s^l,\lambda^l,\mu^l)^\top d^s + \frac{1}{2}d^{s\top}\nabla_{ss}\mathcal{L}_a^k(x^l,s^l,\lambda^l,\mu^l)d^s \\ \text{sujet à} & \left\|x^l + d^x - x^k\right\|_\infty \leq \delta^k, \\ & s^l + d^s \geq 0, \\ & \|d^x\|_\infty \leq \gamma^l, \\ & \|d^s\|_\infty \leq \gamma^l \end{array} \tag{4.25}$$

avec γ^l la région de confiance du modèle. Pour simplifier ce modèle, soit $d \in \mathbb{R}^{n+m}$ la concaté-

Algorithme 9: Lagrangien augmenté pour résoudre (4.23) (itération externe) [10]

1: Initialiser : $l \leftarrow 0,\ \lambda^l \leftarrow 0,\ \mu^l \leftarrow 1.2,\ \eta^l \leftarrow 1,$
$y^{l^\top} = [x^{l^\top} s^{l^\top}],\ \tau \leftarrow 10^{-5},\ \omega^l \leftarrow 1$
2: **Si** $\left\| y^l - \mathcal{P}_{\mathcal{X}\times\mathbb{R}^m_+}(y^l - \nabla_y \mathcal{L}(x^l, s^l, \lambda^l)) \right\| \leq \tau$ et $\left\| c(x^l) + s^l \right\| \leq \tau$
3: Arrêt
4: **Sinon**
5: Aller à **7**
6: **fin Si**
7: Évaluer l'itéré suivant (x^{l+1}, s^{l+1}) en résolvant (4.24) pour des valeurs fixées de λ^l, μ^l et ω^l (Algorithme 11)
8: **Si** $\left\| c(x^{l+1}) + s^{l+1} \right\|_\infty \leq \eta^l$
9: $\lambda^{l+1} \leftarrow \lambda^l - \frac{1}{\mu^l}(c^k(x^l) + s^l)$
10: $\mu^{l+1} \leftarrow \mu^l$
11: $\eta^{l+1} \leftarrow \eta^l (\mu^l)^{0.9}$
12: $\omega^{l+1} \leftarrow \omega^l \mu^{l+1}$
13: **Sinon**
14: $\lambda^{l+1} \leftarrow \lambda^l$
15: $\mu_{l+1} \leftarrow \frac{\mu_l}{10}$
16: $\eta^{l+1} \leftarrow \frac{(\mu^{l+1})^{0.1}}{10}$
17: $\omega^{l+1} \leftarrow \mu^{l+1}$
18: **fin Si**
19: $l \leftarrow l + 1$
20: Aller à **2**

nation de d^x de d^s ($d^\top = [d^{x\top} d^{s\top}]$) et $y \in \mathbb{R}^{n+m}$ la concaténation de x et s ($y^\top = [x^\top s^\top]$). Les contraintes de (4.25) consistent en l'intersection d'une boite définie par les bornes simples avec la boite définie par la région de confiance. Il est donc possible de redéfinir l'ensemble réalisable comme une boite $\mathcal{B}$ avec des bornes inférieures $l \in \mathbb{R}^{n+m}$ et supérieures $u \in \mathbb{R}^{n+m}$:

$$\mathcal{B} = \quad \{d \in \mathbb{R}^{n+m} : l \leq d \leq u\}. \tag{4.26}$$

Le problème (4.25) devient alors :

$$\min_{d\in\mathcal{B}} \quad \nabla_y \mathcal{L}_a^k(y^l, \lambda^l, \mu^l)^\top d + \tfrac{1}{2} d^\top \nabla_{yy} \mathcal{L}_a^k(y^l, \lambda^l, \mu^l) d. \tag{4.27}$$

La résolution du problème (4.27) se base sur une méthode itérative établie par Moré et Toraldo pour les problèmes quadratiques avec contraintes de bornes (voir [10, 130]). Cette méthode était une option pour LANCELOT B dans la librairie GALAHAD [79]. Nous avons choisi cette approche suite à quelques résultats préliminaires sur des problèmes sans contraintes qui favorisent cette approche

de région de confiance par rapport à la méthode de recherche linéaire utilisée dans la section précédente. La méthode de Moré et Toraldo manipule l'ensemble actif $\mathcal{A}(\tilde{d}) = \{i \in \{1, 2, \ldots, n+m\} : \tilde{d}_i = \ell_i \text{ ou } \tilde{d}_i = u_i\}$ pour trouver une direction de descente p utilisée pour mettre à jour l'itéré courant $\tilde{d}$ via recherche linéaire. Dans un premier temps, l'algorithme sélectionne une face prometteuse avec une méthode de gradient projeté. Une face de $\mathcal{B}$ est définie comme un ensemble de vecteurs pour lesquelles les indices de $\mathcal{A}(\tilde{d})$ coïncident avec $\tilde{d}$:

$$\mathcal{F} = \quad \{w \in \mathcal{B} : w_i = \tilde{d}_i \text{ pour } i \in \mathcal{A}(\tilde{d})\}. \tag{4.28}$$

Une fois la face sélectionnée, la méthode de Moré et Toraldo utilise l'algorithme du gradient conjugué pour trouver une direction de descente p dans la face et mettre à jour $\tilde{d}$. L'algorithme 10 résume la méthode de Moré et Toraldo pour trouver la direction de descente d.

Algorithme 10: Algorithme GP/GC [130]

1: **Tant que** pas de solution d trouvée
2: Utiliser la méthode du gradient projeté (PG) pour sélectionner une face prometteuse de l'ensemble réalisable $\mathcal{B}$
3: Utiliser la méthode du gradient conjugué (CG) pour explorer la face sélectionnée et trouver une direction de descente p
4: Utiliser une recherche linéaire projetée le long de p pour choisir l'itéré suivant d
5: **fin Tant que**

L'itération interne du Lagrangien augmenté se base sur une méthode de région de confiance : le ratio r (défini à la ligne 8 de l'algorithme 11) détermine si le modèle est convenable pour la fonction du Lagrangien augmenté. Si le ratio est proche de 1, le modèle est jugé convenable et le rayon de la région de confiance est augmenté. Cependant, si r est négatif ou proche de 0, le rayon est diminué. Le ratio r permet aussi de décider si l'itéré x doit être mis à jour ou pas. L'algorihtme 11 décrit l'itération interne de la méthode du Lagrangien augmenté, avec $\mathcal{L}^k(x, s, \lambda) = f^k(x) - \sum_{j=1}^{m} \lambda_j (c_j^k(x) + s_j)$ et $\mathcal{P}_{\mathcal{X} \times \mathbb{R}_+^m}(y)$ la projection de y sur l'ensemble $\mathcal{X} \times \mathbb{R}_+^m = \{(x, s) : x \in \mathcal{X}, s \in \mathbb{R}_+^m\}$.

4.3 Lagrangien augmenté en norme ℓ_1 (ℓ_1AUGLAG)

Dans cette section, nous proposons une nouvelle méthode qui combine les algorithmes des deux sections précédentes et qui est explorée en détail dans le chapitre 5. Comme le problème (4.3) est quadratique, le Lagrangien augmenté perd la structure quadratique du sous-problème à cause du terme de pénalité quadratique. Nous proposons de remplacer ce terme par une pénalité en norme ℓ_1 pour avoir une fonction quadratique par morceaux. Nous définissons le problème du Lagrangien augmenté en norme ℓ_1 de la manière suivante :

Algorithme 11: Lagrangien augmenté (itération interne) [10]

1: Initialisations : d, $\epsilon_1 \leftarrow 0.1,\ \epsilon_2 \leftarrow 0.9, \gamma_1 \leftarrow 0.01,\ \gamma_2 \leftarrow 100$
2: **Si** $\left\| y^l - \mathcal{P}_{\mathcal{X}\times\mathbb{R}^m_+}(y^l - \nabla_y \mathcal{L}(x^l, s^l, \lambda^l)) \right\| \leq \omega^l$
3: Arrêt
4: **Sinon**
5: Aller à **7**
6: **fin Si**
7: Résoudre (4.27) pour trouver d (Algorithme 10)
8: Calculer le ratio $r = \dfrac{\mathcal{L}_a^k(y^l + d, \lambda^l, \mu^l) - \mathcal{L}_a^k(y^l, \lambda^l, \mu^l)}{\nabla \mathcal{L}_a^k(y^l, \lambda^l, \mu^l)^\top d + \frac{1}{2} d^\top \nabla^2 \mathcal{L}_a^k(y^l, \lambda^l, \mu^l) d}$
9: **Si** $r \geq \epsilon_1$
10: $x^l \leftarrow x^l + d$
11: **Si** $r \geq \epsilon_2$
12: $\gamma \leftarrow \gamma_2 \max\{\|d\|_\infty, \gamma\}$
13: **fin Si**
14: **Sinon**
15: $\gamma \leftarrow \gamma_1 \|d\|_\infty$
16: **fin Si**
17: Aller à **2**

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f^k(x) - \sum\limits_{j=1}^m \lambda_j c_j^k(x) + \frac{1}{\mu} \sum\limits_{j=1}^m \max\{0, c_j^k(x)\} \\ \text{sujet à} & \left\| x - x^k \right\|_\infty \leq \delta^k. \end{array} \tag{4.29}$$

En utilisant les mêmes définitions des ensembles ϵ-actives et ϵ-violées que l'équation (4.5) pour un point $\tilde{x} = x^l$ (à l'itération l de la boucle externe), il est possible de limiter l'étude du problème (4.29) dans le voisinage $\mathcal{N}(x^l) \subseteq \mathcal{X}$ de x^l (même approche que dans (4.6)) :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{N}(x^l)} & D^k(x, \lambda, \mu) + \frac{1}{\mu} \sum\limits_{j\in A_\epsilon} \max\{0, c_j^k(x)\} \\ \text{sujet à} & \left\| x - x^k \right\|_\infty \leq \delta^k \end{array} \tag{4.30}$$

où $D^k(x, \lambda, \mu) = f^k(x) - \sum\limits_{j=1}^m \lambda_j c_j^k(x) + \frac{1}{\mu} \sum\limits_{j\in V_\epsilon} c_j^k(x)$ est différentiable, mais le second terme pénalisant les contraintes ϵ-active ne l'est pas. Ceci est similaire à l'approche précédente de la fonction de pénalité exacte en norme ℓ_1 : à chaque itération l, le traitement des problèmes de différentiabilité se fait en se concentrant sur l'évaluation d'une direction de descente qui minimise la variation de la fonction du Lagrangien augmenté en norme ℓ_1 tout en empêchant la variation des contraintes ϵ-actives pour des valeurs fixées de λ et μ :

$$\begin{array}{ll} \min\limits_{h\in\mathbb{R}^n} & \nabla_x D^k(x^l,\lambda^l,\mu^l)^\top h + \frac{1}{2}h^\top \nabla_{xx} D^k(x^l,\lambda^l,\mu^l)h \\ \text{sujet à} & \nabla c_j^k(x^l)^\top h + \frac{1}{2}h^\top \nabla^2 c_j^k(x^l)h = 0, \qquad j \in A_\epsilon. \end{array} \tag{4.31}$$

Pour satisfaire les contraintes ϵ-actives, il est possible de calculer une direction de descente v en utilisant l'équation (4.17). L'algorithme 6 reste valide en obtenant la direction de descente h dans les lignes 5 et 13 par la résolution du problème (4.31) cette fois-ci. Pour la mise à jour des multiplicateurs λ et du terme de pénalité μ, nous utilisons une approche similaire à la boucle externe de l'algorithme du Lagrangien augmenté : le paramètre μ peut être mis à jour de la même manière, mais pour λ, des ajustements sont nécessaires. Pour cela, nous nous basons sur le test d'optimalité suivant :

$$\nabla_x \mathcal{L}^k(x^l,\lambda^l) = 0 \tag{4.32}$$

où $\mathcal{L}^k(x,\lambda) = f^k(x) - \sum\limits_{j=1}^{m} \lambda_j c_j^k(x)$ est le Lagrangien du problème (4.3). Donc, à l'itération l, si x^{l+1} est un minimum du problème (4.30), il existe un vecteur $\bar{\lambda}$ tel que :

$$\nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j^k(x^{l+1}) + \frac{1}{\mu}\sum_{j\in V_\epsilon} \nabla c_j^k(x^{l+1}) - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x^{l+1}) = 0. \tag{4.33}$$

Si nous adoptons les règles de mise à jour suivante pour les multiplicateurs λ :

$$\lambda^{l+1} = \begin{cases} \lambda_j^l + \bar{\lambda}_j, & \text{si } j \in A_\epsilon, \\ \lambda_j^l - \frac{1}{\mu}, & \text{si } j \in V_\epsilon, \\ \lambda_j^l, & \text{si } j \notin A_\epsilon \cup V_\epsilon, \end{cases} \tag{4.34}$$

alors le test d'optimalité (4.32) donne :

$$\begin{aligned} \nabla_x \mathcal{L}^k(x^{l+1},\lambda^{l+1}) &= \nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j^{l+1} \nabla c_j^k(x^{l+1}) \\ &= \nabla f^k(x^{l+1}) - \sum_{j\notin A_\epsilon\cup V_\epsilon} \lambda_j^l \nabla c_j^k(x^{l+1}) \\ &\quad - \sum_{j\in V_\epsilon} (\lambda_j^l - \tfrac{1}{\mu})\nabla c_j^k(x^{l+1}) - \sum_{j\in A_\epsilon} (\lambda_j^l + \bar{\lambda}_j)\nabla c_j^k(x^{l+1}) \\ &= \nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j^k(x^{l+1}) + \tfrac{1}{\mu}\sum_{j\in V_\epsilon} \nabla c_j^k(x^{l+1}) \\ &\quad - \sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x^{l+1}) \\ &= 0 \end{aligned} \tag{4.35}$$

ce qui confirme les règles de mise à jour adoptées. Pour évaluer $\bar{\lambda}$, nous réécrivons l'équation (4.33)

de la manière suivante :

$$\sum_{j\in A_\epsilon} \bar{\lambda}_j \nabla c_j^k(x^{l+1}) = \nabla f^k(x^{l+1}) - \sum_{j=1}^{m} \lambda_j \nabla c_j^k(x^{l+1}) + \frac{1}{\mu} \sum_{j\in V_\epsilon} \nabla c_j^k(x^{l+1}). \tag{4.36}$$

En utilisant une notation matricielle, le système (4.36) devient :

$$J_{A_\epsilon}^k \bar{\lambda} = \nabla D^k(x^{l+1}, \lambda^{l+1}, \mu^{l+1}) \tag{4.37}$$

avec $J_{A_\epsilon}^k$ la matrice des gradients des contraintes ϵ-actives définie dans la section 4.1. En utilisant la décomposition QR, comme précédemment, il existe une matrice orthogonale Q et une matrice triangulaire supérieure U telles que :

$$J_{A_\epsilon}^k = QR = [Q_1\ Q_2] \begin{bmatrix} U \\ 0 \end{bmatrix} = Q_1 U \tag{4.38}$$

où le nombre de colonnes de la matrice Q_1 est égal au nombre de ligne de la matrice U et correspond au nombre des contraintes ϵ-actives à l'itéré courant. L'équation (4.37) devient alors :

$$U\ \bar{\lambda} = Q_1^\top \nabla D^k(x^{l+1}, \lambda^{l+1}, \mu^{l+1}). \tag{4.39}$$

Il est maintenant facile d'obtenir $\bar{\lambda}$ vu que la matrice U est triangulaire supérieure. L'algorithme 12 résume la boucle externe de l'algorithme ℓ_1AUGLAG (avec $\mathcal{L}^k(x,\lambda) = f^k(x) - \sum_{j=1}^{m} \lambda_j c_j^k(x)$ et $\mathcal{P}_\mathcal{X}(y)$ la projection de y sur l'ensemble $\mathcal{X}$).

4.4 Résultats numériques

Nous utilisons trois ensembles de problèmes pour comparer entre cinq variantes de NOMAD : chaque version résout le sous-problème (4.3) différemment.

4.4.1 Problèmes tests

Le premier ensemble contient 42 problèmes analytiques sans contraintes, le second ensemble se compose de 19 problèmes analytiques contraints et le troisième groupe consiste en 4 problèmes contraints basés sur des simulations : deux applications MDO appelées AIRCRAFT_RANGE et SIMPLIFIED_WING, un problème d'ingénierie de conception structurelle d'une poutre soudée nommé WELDED et une boite noire (LOCKWOOD) qui minimise le coût de gestion des puits

Algorithme 12: Lagrangien augmenté en norme l_1 : Boucle externe

1: Initialiser : $l \leftarrow 0,\ \lambda^l \leftarrow 0,\ \mu^l \leftarrow 1,\ \eta^l \leftarrow 1,$
$x^l,\ \tau \leftarrow 10^{-5},\ \omega^l \leftarrow 1$
2: **Si** $\left\| x^l - \mathcal{P}_{\mathcal{X}}(x^l - \nabla_x \mathcal{L}(x^l, \lambda^l)) \right\| \leq \tau$
3: Arrêt
4: **Sinon**
5: Aller à **7**
6: **fin Si**
7: Trouver l'itéré suivant x^{l+1} en minimisant (4.30) pour des valeurs fixes de λ^l, μ^l et ω^l
8: **Si** $c_j^k(x^{l+1}) \leq \eta^l$ pour tout $j \in V_\epsilon$
9: $\lambda^{l+1} \leftarrow \begin{cases} \lambda_j^l + \bar{\lambda_j} & j \in A_\epsilon \\ \lambda_j^l - \frac{1}{\mu} & j \in V_\epsilon \\ \lambda_j^l & j \notin A_\epsilon \cup V_\epsilon \end{cases}$
10: $\mu^{l+1} \leftarrow \mu^l$
11: $\eta^{l+1} \leftarrow \eta^l (\mu^l)^{0.9}$
12: $\omega^{l+1} \leftarrow \omega^l \mu^{l+1}$
13: **Sinon**
14: $\lambda^{l+1} \leftarrow \lambda^l$
15: $\mu_{l+1} \leftarrow \frac{\mu_l}{10}$
16: $\eta^{l+1} \leftarrow \frac{(\mu^{l+1})^{0.1}}{10}$
17: $\omega^{l+1} \leftarrow \mu^{l+1}$
18: **fin Si**
19: $l \leftarrow l + 1$
20: Aller à **2**

dans le site d'eau souterraine Lockwood (Montana) qui empêche la contamination de la rivière Yellowstone. Le tableau 4.1 regroupe le nom, la dimension, le nombre de contraintes et la source de ces 65 problèmes tests.

Nous avons implémenté les méthodes décrites dans les sections 4.1, 4.2 et 4.3 dans le logiciel NOMAD [109] version 3.7 (`www.gerad.ca/nomad`) et nous comparons cinq variantes algorithmiques. La première est dénotée MADS et consiste à utiliser l'algorithme de recherche directe sur treillis adaptatifs pour résoudre le sous-problème (4.3). Trois autres variantes ℓ_1EPF, AUGLAG et ℓ_1AUGLAG utilisent respectivement la fonction de pénalité exacte en norme ℓ_1, le Lagrangien augmenté et le Lagrangien augmenté en norme ℓ_1 pour traiter le sous-problème (4.3). Finalement, la dernière version utilise le logiciel IPOPT [163] qui est conçu pour l'optimisation non linéaire. Les tests sont effectués sur une machine Linux avec un processeur Intel(R) Core(TM) i7-2600 (3.40 GHz).

Les résultats des tests numériques sont présentés par l'intermédiaire de profils de données et de performance [57, 131] (voir aussi l'annexe A du livre [22]). Si $\mathcal{S}$ est l'ensemble des méthodes à

Tableau 4.1 Description des problèmes tests.

Nom	n	m	Source	Nom	n	m	Source	Nom	n	m	Source
Problèmes sans contraintes								Problèmes avec contraintes			
ACKLEY	10	0	[91]	POWELLSG	12	0	[78]	CRESCENT10	10	2	[19]
ARWHEAD	10	0	[78]	POWELLSG	20	0	[78]	DISK10	10	1	[19]
ARWHEAD	20	0	[78]	RADAR	7	0	[128]	G1	13	9	[126]
BDQRTIC	10	0	[78]	RANA	2	0	[99]	G2	10	2	[126]
BDQRTIC	20	0	[78]	RASTRIGIN	2	0	[91]	G2	20	2	[126]
BIGGS6	6	0	[78]	RHEOLOGY	3	0	[22]	G4	5	6	[126]
BRANIN	2	0	[91]	ROSENBROCK	2	0	[99]	G6	2	2	[126]
BROWNAL	10	0	[78]	SCHWEFEL	2	0	[150]	G7	10	8	[126]
BROWNAL	20	0	[78]	SHOR	5	0	[118]	G8	2	2	[126]
DIFF2	2	0	[49]	SROSENBR	10	0	[78]	G9	7	4	[126]
ELATTAR	6	0	[118]	SROSENBR	20	0	[78]	G10	8	6	[126]
EVD61	6	0	[118]	TREFETHEN	2	0	[99]	HS44	4	6	[94]
FILTER	9	0	[118]	TRIDIA	10	0	[78]	HS64	3	1	[94]
GRIEWANK	2	0	[91]	TRIDIA	20	0	[78]	HS93	6	2	[94]
GRIEWANK	10	0	[91]	VARDIM	10	0	[78]	HS108	9	13	[94]
HS78	5	0	[118]	VARDIM	20	0	[78]	HS114	9	6	[118]
OSBORNE2	11	0	[118]	WATSON	12	0	[99]	MAD6	5	7	[118]
PBC1	5	0	[118]	WONG1	7	0	[118]	PENTAGON	6	15	[118]
PENALTY1	10	0	[78]	WONG2	10	0	[118]	SNAKE	2	2	[19]
PENALTY1	20	0	[78]	WOODS	12	0	[78]				
POLAK2	10	0	[118]	WOODS	20	0	[78]				
Applications MDO basées sur des simulations											
AIRCRAFT_RANGE	10	10	[6, 154]	SIMPLIFIED_WING	7	3	[162]	WELDED	4	6	[72]
LOCKWOOD	6	4	[103, 122]								

comparer et $\mathcal{P}$ est l'ensemble d'instances tests, la mesure de performance $t_{p,s}$, pour $p \in \mathcal{P}$ et $s \in \mathcal{S}$, représente le nombre d'évaluations minimum nécessaire pour satisfaire le test de convergence suivant :

$$f(x_{feas}) - f(x) \geq (1-\tau)(f(x_{feas}) - f_L), \tag{4.40}$$

où τ est la précision de convergence, f_L est la meilleure solution trouvée par tous les algorithmes $s \in \mathcal{S}$ pour un problème donné et un budget fixé (un budget d'évaluation ou de temps) et x_{feas} est le premier point réalisable trouvé par n'importe quelle méthode. Si le point de départ x_0 d'un problème est réalisable, alors $x_{feas} = x_0$. Si un algorithme n'arrive pas à trouver un point réalisable pour un problème donné, le test de convergence échoue et ceci favorise tout algorithme qui trouve au moins une solution réalisable. Il est possible de définir x_{feas} différemment. Par exemple, dans [27], x_{feas} représente la moyenne de la première valeur réalisable de la fonction objectif de toutes les instances d'un problème.

Pour une méthode $s \in \mathcal{S}$, le profil de données est défini par :

$$d_s(\kappa) = \frac{1}{card(\mathcal{P})} card\left\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa\right\}, \tag{4.41}$$

où n_p est le nombre de variables du problème p. Le profil de données représente le ratio d'instances

résolues par s pour une tolérance τ et pour un nombre κ de groupes de $n_p + 1$ évaluations. La comparaison entre des profils de données permet de déterminer le meilleur algorithme pour un budget fixé.

Pour décrire les profils de performance, il faut d'abord définir, pour $p \in \mathcal{P}$ et $s \in \mathcal{S}$, le ratio de performance $r_{p,s}$:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,u} : u \in \mathcal{S}\}}. \tag{4.42}$$

Pour un algorithme $s \in \mathcal{S}$, le profil de performance est défini par :

$$\rho_s(\alpha) = \frac{1}{card(\mathcal{P})} card\left\{p \in \mathcal{P} : r_{p,s} \leq \alpha\right\}, \tag{4.43}$$

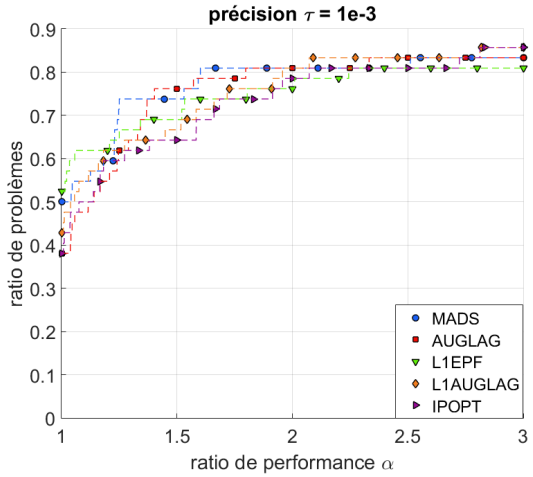
Un profil de performance donne le pourcentage de problèmes résolus vis-à-vis d'un ratio de performance α défini comme la borne supérieure sur le temps d'exécution ou le nombre d'évaluations nécessaire pour satisfaire le test de convergence. Ces profils permettent de comparer les performances relatives des différents algorithmes.

4.4.2 Problèmes sans contraintes

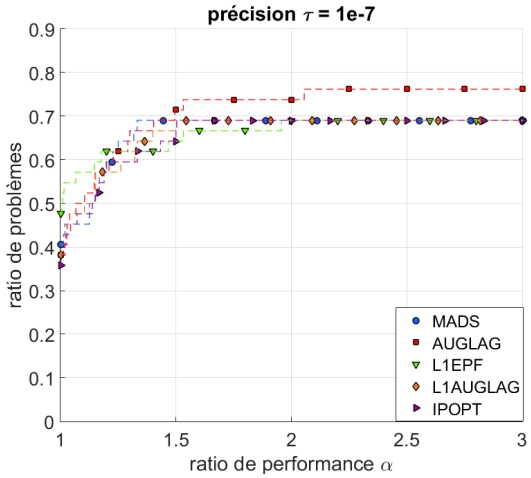
Comme les méthodes utilisées se basent sur des multiplicateurs et termes de pénalité, il peut paraître étrange de tester des problèmes sans contraintes, cependant, dans ce cas, ces méthodes se réduisent à leur boucle interne : les approches ℓ_1EPF et ℓ_1AUGLAG deviennent des algorithmes de recherche linéaire qui utilisent des décompositions matricielles pour sélectionner des directions de descente, alors que la méthode AUGLAG se transforme en un algorithme de région de confiance qui se base sur une combinaison des approches de gradients projeté et conjugué. Nous avons opté à présenter le cas non contraint vu que les résultats nous mènent à recommander une approche différente par rapport au cas avec contraintes.

Il est difficile de discerner un vainqueur parmi les cinq variantes algorithmiques dans la figure 4.2(a) qui regroupe les profils de performances pour l'ensemble non contraint avec une précision $\tau = 10^{-3}$. Par contre dans les profils de performance 4.2(b) et de données 4.2(c) pour le même ensemble de problèmes avec une précision $\tau = 10^{-7}$, AUGLAG l'emporte clairement par rapport aux autres méthodes.

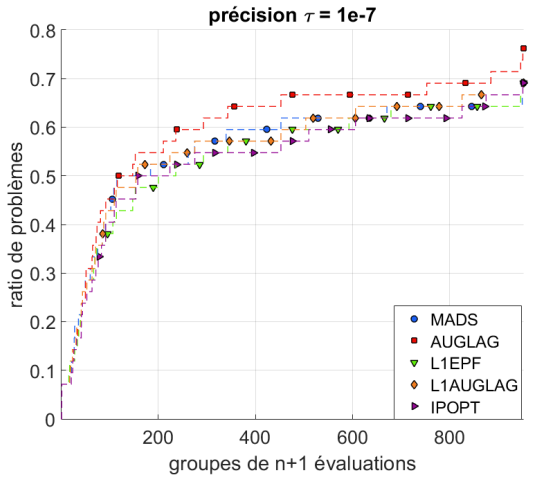
La figure 4.2(d) donne les profils de données temporels qui représentent le ratio de problèmes résolus pour un budget de temps donné avec une précision $\tau = 10^{-7}$. Cette figure prend en considération l'effort fourni pour résoudre les sous-problèmes quadratiques et confirme la supériorité de



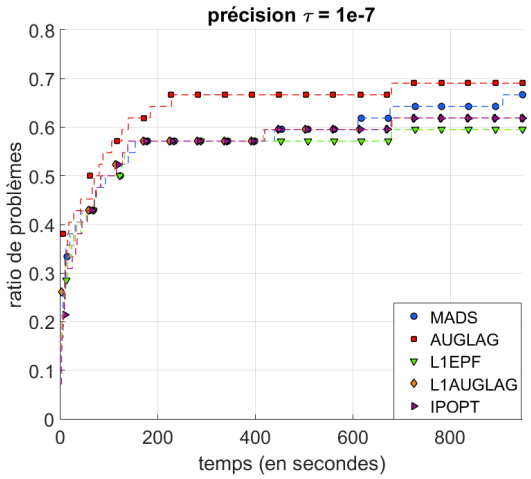
(a) Profils de performance pour $\tau = 10^{-3}$.



(b) Profils de performance pour $\tau = 10^{-7}$.



(c) Profils de données pour $\tau = 10^{-7}$.



(d) Profils de données temporelles pour $\tau = 10^{-7}$.

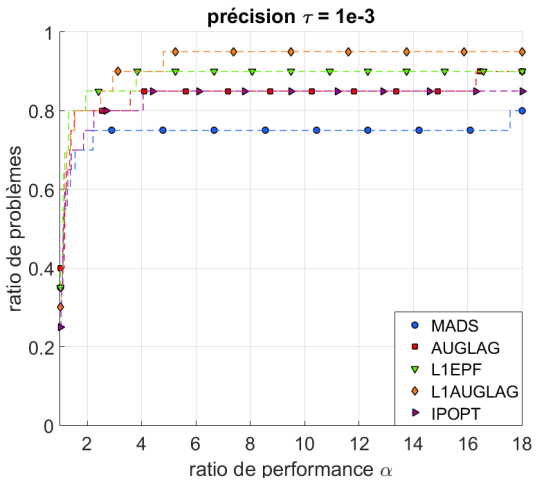
Figure 4.2 Profils de performance et de données pour le cas sans contraintes.

la méthode de Moré et Toraldo dans le cas sans contraintes.

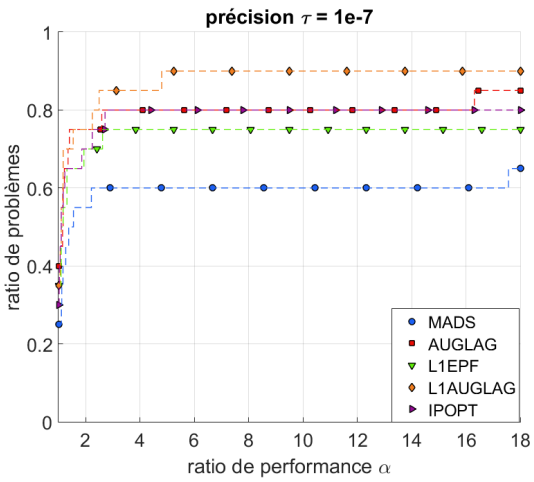
Les différences dans les profils entre ℓ_1EPF et ℓ_1AUGLAG, qui se réduisent à la même boucle interne, sont dues à des choix différents du point de départ pour les sous-problèmes. Ce choix d'implémentation provient de tests préliminaires sur des problèmes contraints pour sélectionner le meilleur point de départ pour chacune des trois méthodes. Nous avons décidé de garder ce choix pour le cas sans contraintes. Les deux approches semblent quasiment équivalentes vu que les différences entre les deux méthodes dans les profils sont mineures.

Nous concluons que la méthode de région de confiance est préférable à la recherche linéaire pour traiter les sous-problèmes sans contraintes qui surgissent lors d'utilisation de modèles quadratiques dans MADS.

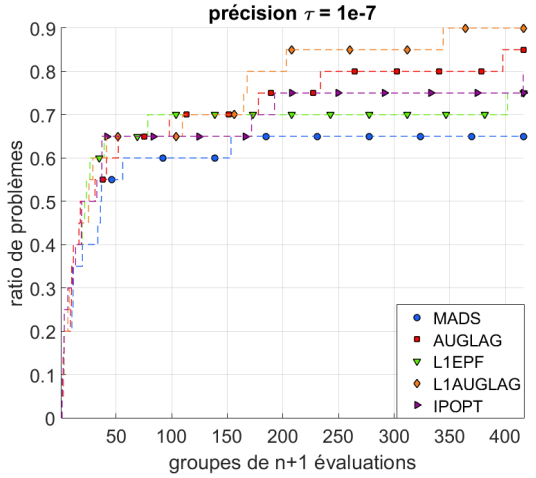
4.4.3 Problèmes avec contraintes



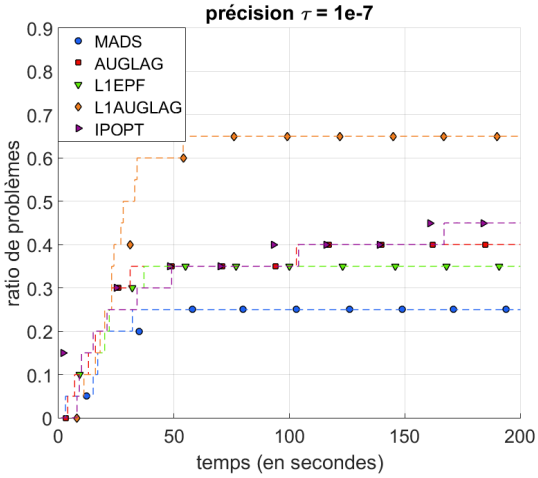
(a) Profils de performance pour $\tau = 10^{-3}$.



(b) Profils de performance pour $\tau = 10^{-7}$.



(c) Profils de données pour $\tau = 10^{-7}$.



(d) Profils de performance et de données pour le cas sans contraintes.

Figure 4.3 Profils de performance et de données pour le cas avec contraintes.

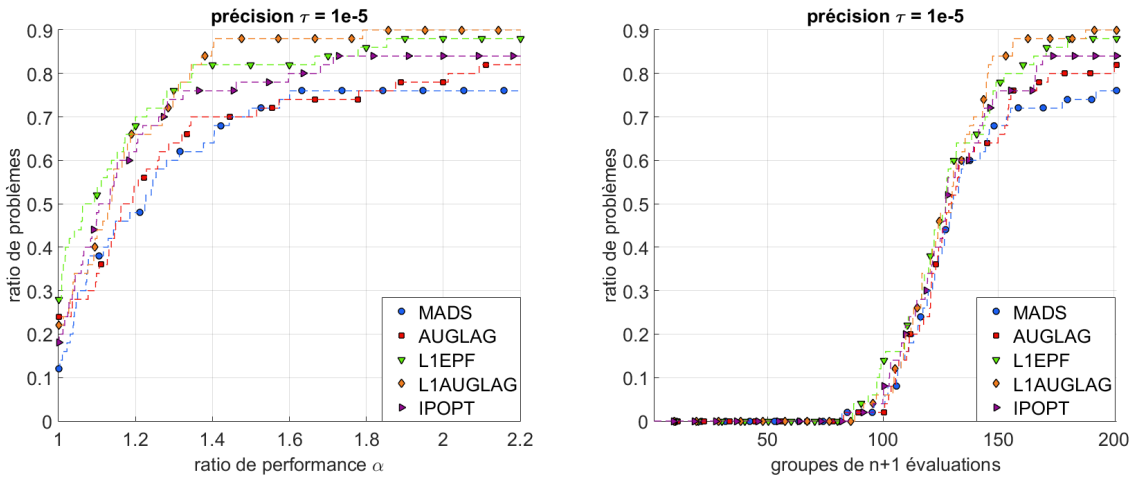
Le second ensemble regroupe les problèmes avec contraintes. La figure 4.3 dresse les profils de performance et de données pour les précisions $\tau = 10^{-3}$ et $\tau = 10^{-7}$. Pour cet ensemble de problèmes contraints, toutes les nouvelles méthodes surpassent MADS. Plus spécifiquement, la variante ℓ_1AUGLAG donne les meilleurs résultats et arrive à résoudre quasiment 90% des problèmes alors que MADS s'arrête à 60% pour $\tau = 10^{-7}$, comme le montrent les profils de performance 4.3(b).

Les profils de données temporels dans la figure 4.3(d) montrent que IPOPT est efficace, mais que ℓ_1AUGLAG reste la meilleure approche pour les problèmes avec contraintes.

4.4.4 Simulations réelles

Pour le dernier ensemble test, nous créons 50 instances différentes pour chacune des quatre applications : pour chaque problème, l'échantillonnage par hypercube latin [124] génère 50 points de départ et chacune de ces instances est traitée par les cinq variantes de NOMAD. Les points de départ générés peuvent être réalisables ou pas. Ces simulations sont souvent temporellement coûteuses : les 750 tests de AIRCRAFT_RANGE, SIMPLIFIED_WING et WELDED ont pris un peu plus de cinq jours d'exécution et les 250 tests de LOCKWOOD ont pris quasiment deux semaines de temps d'exécution.

Pour chaque application, nous dressons les profils de performance et de données des différentes variantes algorithmiques en considérant que toutes les instances sont des problèmes différents. Pour chaque problème, il est important de bien sélectionner une précision τ appropriée, car, d'un côté, si la valeur de τ est trop petite, les profils ont tendance à s'entasser avec un faible ratio de problèmes résolus. D'un autre côté, si la valeur de τ est trop grande, les courbes convergent très rapidement à 100% de problèmes résolus. Pour la suite, nous choisissons les valeurs de τ comme la puissance de 10 qui disperse le plus possible les profils.



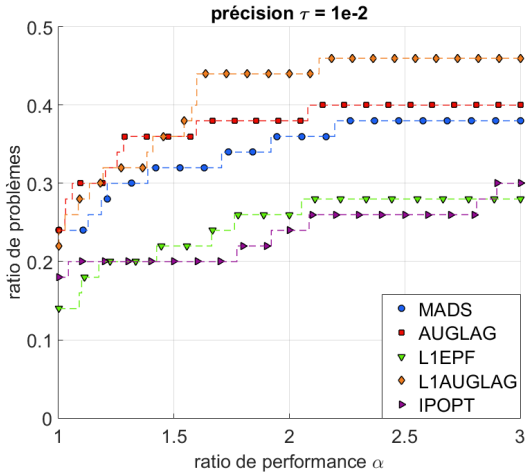
(a) Profils de performance pour $\tau = 10^{-5}$.

(b) Profils de données pour $\tau = 10^{-5}$.

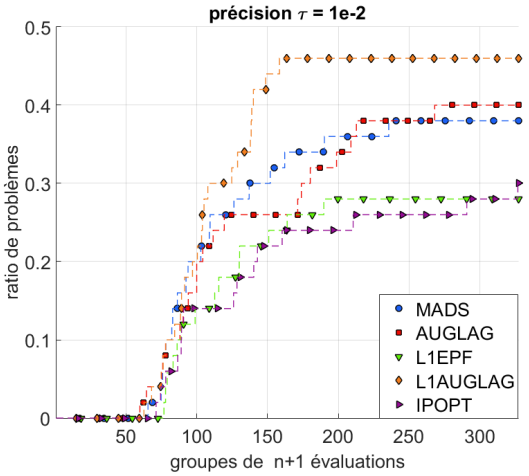
Figure 4.4 Profils de performance et de données pour le problème AIRCRAFT_RANGE.

Dans les figures 4.4 et 4.5, ℓ_1AUGLAG paraît comme la meilleure approche pour les applications AIRCRAFT_RANGE et LOCKWOOD. Mais, si ℓ_1EPF et IPOPT donnent de bons résultats pour l'application AIRCRAFT_RANGE (il y'a même un chevauchement avec ℓ_1AUGLAG pour $\alpha \leq 1.3$), les variantes AUGLAG et MADS sont plus efficaces pour la boite noire LOCKWOOD.

Dans la figure 4.6, MADS donne de bons résultats quand $\alpha < 4.5$ et le budget est limité à $300(n+1)$ évaluations. Cependant, si plus de budget est alloué, ℓ_1AUGLAG peut quasiment atteindre 80% de

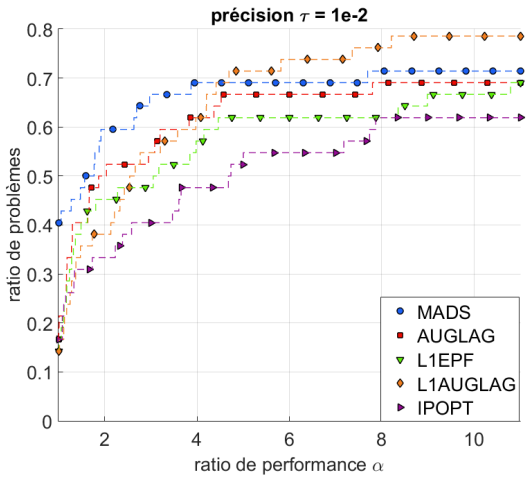


(a) Profils de performance pour $\tau = 10^{-2}$.

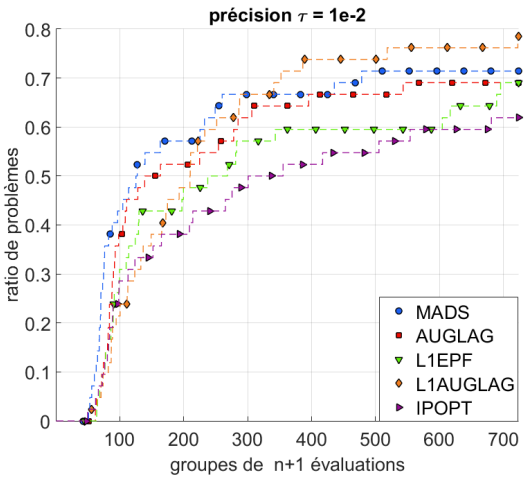


(b) Profils de données pour $\tau = 10^{-2}$.

Figure 4.5 Profils de performance et de données pour le problème LOCKWOOD.



(a) Profils de performance pour $\tau = 10^{-2}$.



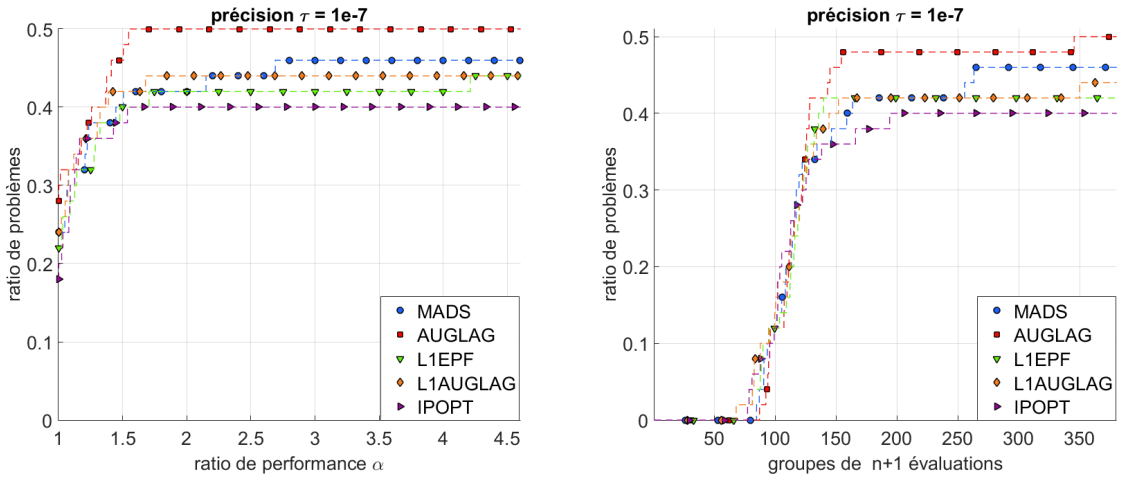
(b) Profils de données pour $\tau = 10^{-2}$.

Figure 4.6 Profils de performance et de données pour le problème SIMPLIFIED_WING.

problèmes résolus.

Dans la figure 4.7, l'approche AUGLAG surpasse le reste des algorithmes avec IPOPT finissant encore une fois en dernière position. Nous ne savons pas pourquoi AUGLAG est meilleur pour cette application, mais ceci montre que ℓ_1AUGLAG ne gagne pas tout le temps.

Dans tous les graphes des applications réelles, IPOPT fonctionne assez mal comparé aux autres méthodes, excepté pour les instances de AIRCRAFT_RANGE. Pour recommander une alternative à MADS parmi les trois nouvelles méthodes, nous dressons les profils de performances et de don-



(a) Profils de performance pour $\tau = 10^{-7}$.

(b) Profils de données pour $\tau = 10^{-7}$.

Figure 4.7 Profils de performance et de données pour le problème WELDED.

nées dans la figure 4.8 avec $\tau = 10^{-2}$ et $\tau = 10^{-5}$ pour la totalité des 200 instances des quatre applications réelles.

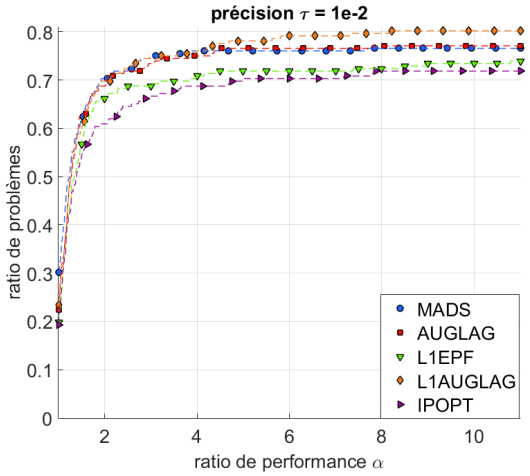
La figure 4.8 montre que ℓ_1AUGLAG surpasse toutes les autres variantes. La méthode se basant sur IPOPT n'est pas une option à considérer pour résoudre les sous-problèmes quadratiques dans MADS. Ceci était prévisible vu que IPOPT est utilisé pour des problèmes d'optimisation non linéaires généraux alors que des méthodes comme ℓ_1AUGLAG sont plus adaptées aux problèmes quadratiques avec contraintes quadratiques.

4.5 Discussion

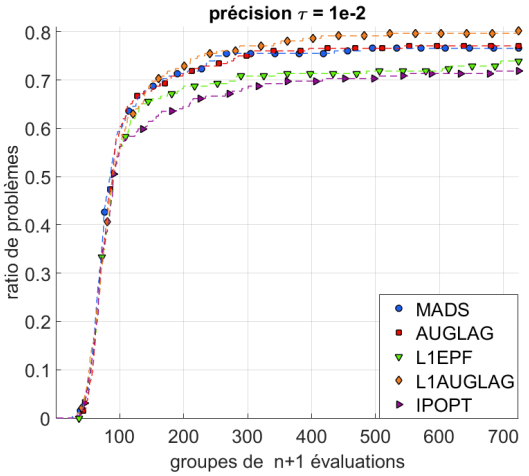
Ce chapitre teste l'intégration de deux algorithmes classiques d'optimisation non linéaire (ℓ_1EPF et AUGLAG) et d'une nouvelle méthode nommée Lagrangien augmenté en norme ℓ_1 (ℓ_1AUGLAG) dans un cadre d'optimisation sans dérivées pour résoudre les sous-problèmes quadratiques avec contraintes quadratiques.

Cette étude démontre que l'utilisation d'un algorithme dédié pour résoudre les sous-problèmes quadratiques améliore la performance de MADS lors de l'utilisation de modèles quadratiques dans la phase de recherche. Les tests numériques montrent que, dans le cas sans contraintes, l'utilisation d'une méthode de région de confiance (itération interne de AUGLAG basée sur la méthode de Moré et Toraldo [130]) donne de meilleurs résultats qu'une approche de recherche linéaire (itération interne de ℓ_1EPF et ℓ_1AUGLAG).

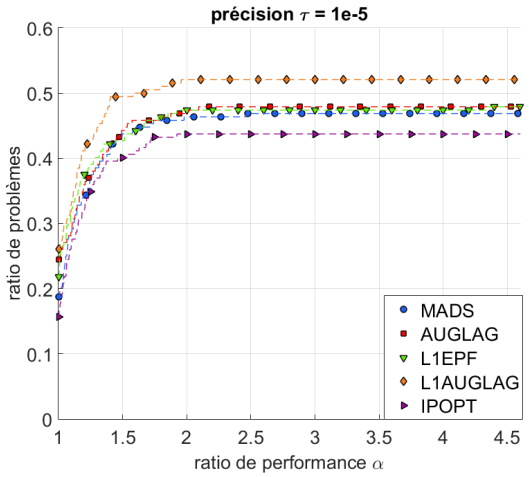
Dans le cas contraint, c'est ℓ_1AUGLAG qui donne les meilleurs résultats vu que cette méthode



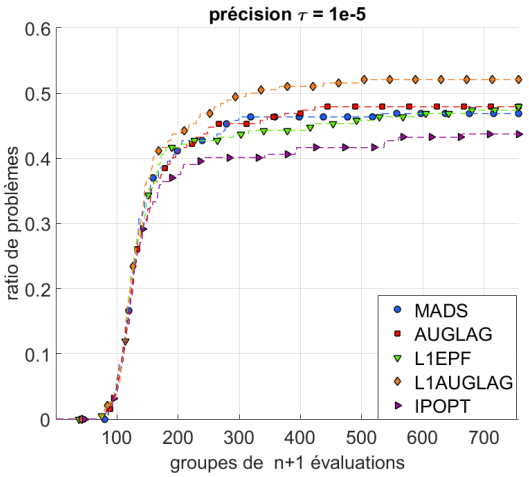
(a) Profils de performance pour $\tau = 10^{-2}$.



(b) Profils de données pour $\tau = 10^{-2}$.



(c) Profils de performance pour $\tau = 10^{-5}$.



(d) Profils de données pour $\tau = 10^{-5}$.

Figure 4.8 Profils de performance et de données pour toutes les 200 instances des applications industrielles.

combine les avantages de AUGLAG et ℓ_1EPF : les multiplicateurs de Lagrange améliorent les performances algorithmiques et le terme de pénalité ℓ_1 permet de garder une structure quadratique par morceaux du problème.

CHAPITRE 5 CONVERGENCE DE L'ALGORITHME DU LAGRANGIEN AUGMENTÉ EN NORME ℓ_1

Le chapitre 4 introduit l'algorithme du Lagrangien augmenté en norme ℓ_1 pour optimiser les sous-problèmes quadratiques qui surgissent lors de l'utilisation de MADS avec des modèles. Dans ce chapitre, nous présentons des résultats de convergence pour cette nouvelle méthode en se basant sur l'analyse de la méthode de Coleman et Conn [42, 43] ainsi que les propriétés de convergence de la méthode du Lagrangien augmenté classique établies dans [44].

Nous sortons donc du contexte de MADS et nous considérons le problème non linéaire avec contraintes d'égalité suivant :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f(x) \\ \text{sujet à} & c_j(x) = 0, \quad j \in \{1, 2, \ldots, m\} \end{array} \tag{5.1}$$

où $\mathcal{X}$ est un sous-ensemble de $\mathbb{R}^n$, f et les c_j, $j \in \{1, 2, \ldots, m\}$, sont des fonctions à valeurs réelles et m et n sont des entiers positifs. Pour optimiser ce problème, nous allons utiliser le Lagrangien augmenté en norme ℓ_1 qui combine les avantages du Lagrangien augmenté classique avec la méthode de la fonction de pénalité exacte en norme ℓ_1. En utilisant cette norme à la place d'une pénalité quadratique, la nouvelle approche reformule le problème en une fonction objectif quadratique par morceaux (dans le cas où le problème (5.1) est un QCQP) alors que le Lagrangien augmenté standard transforme (5.1) en une quartique.

5.1 L'algorithme de Lagrangien augmentée en norme ℓ_1

L'algorithme du Lagrangien augmenté en norme ℓ_1 commence par transformer le problème (5.1) en :

$$\min_{x\in\mathcal{X}} \mathcal{L}_A^{\ell_1}(x, \lambda, \mu) \tag{5.2}$$

où :

$$\mathcal{L}_A^{\ell_1}(x, \lambda, \mu) = f(x) - \sum_{j=1}^{m} \lambda_j c_j(x) + \frac{1}{\mu} \sum_{j=1}^{m} s_{jj} \left| c_j(x) \right| \tag{5.3}$$

avec $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^+$ et les s_{jj}, $j \in \{1, 2, \ldots, m\}$, sont les éléments de la diagonale d'une matrice $S \in \mathbb{R}^{m\times m}$ de mise en échelle des contraintes. Pour surmonter les problèmes de différentiabilité de cette expression, à l'itération k, il faut définir l'ensemble des indices des contraintes ϵ^k-actives et

ϵ^k-violées autour de l'itéré courant $x^k \in \mathcal{X}$ et pour $\epsilon^k > 0$:

$$A_{\epsilon^k}(x^k) = \{j \in \{1, 2, \ldots, m\} : \left|c_j(x^k)\right| \leq \epsilon^k\}, \tag{5.4}$$

$$V_{\epsilon^k}(x^k) = \{j \in \{1, 2, \ldots, m\} : \left|c_j(x^k)\right| > \epsilon^k\}. \tag{5.5}$$

Pour alléger les expressions, on note A_{ϵ^k} à la place de $A_{\epsilon^k}(x^k)$ pour représenter les contraintes ϵ-actives à l'itéré courant x^k. De même, nous notons V_{ϵ^k} à la place de $V_{\epsilon^k}(x^k)$ pour décrire les contraintes ϵ-violées à l'itéré courant x^k. Ces deux définitions permettent de réécrire le problème (5.2) dans un voisinage $\mathcal{N}(x^k)$ de x^k :

$$\min_{x \in \mathcal{N}(x^k) \cap \mathcal{X}} D(x, \lambda^k, \mu^k) + \frac{1}{\mu^k} \sum_{j \in A_{\epsilon^k}} s_{jj}^k \left|c_j(x)\right| \tag{5.6}$$

avec :

$$\begin{aligned} D(x, \lambda^k, \mu^k) &= f(x) - \sum_{j=1}^{m} \lambda_j^k c_j(x) + \tfrac{1}{\mu^k} \sum_{j \in V_{\epsilon^k}} s_{jj}^k \left|c_j(x)\right| \\ &= f(x) - \sum_{j=1}^{m} \lambda_j^k c_j(x) + \tfrac{1}{\mu^k} \sum_{j \in V_{\epsilon^k}} \sigma_j s_{jj}^k c_j(x), \end{aligned} \tag{5.7}$$

où $\sigma_j = sgn(c_j(x))$. La fonction D est différentiable par rapport à x, mais $x \mapsto \frac{1}{\mu^k} \sum_{j \in A_{\epsilon^k}} s_{jj}^k \left|c_j(x)\right|$ ne l'est pas. La méthode se concentre principalement sur l'évaluation d'une direction de descente d qui minimise la variation $\nabla D^\top d$ tout en gardant les contraintes ϵ^k-actives inchangées pour des valeurs fixées de λ^k et μ^k (dans ce chapitre, l'opérateur ∇ représente le gradient par rapport à x). Si l'évaluation d'une telle direction n'est pas possible, alors soit l'itéré courant satisfait les conditions d'optimalité du premier ordre, soit il est possible de relaxer une des contraintes ϵ^k-actives.

Soit $\mathcal{P}$ l'opérateur de projection orthogonale sur le noyau de $\{\nabla c_j(x^k)\}_{j \in A_{\epsilon^k}}$. Si la norme de la projection du gradient de D est non nulle, $\left\|\mathcal{P}\nabla D(x^k, \lambda^k, \mu^k)\right\| \neq 0$, alors il est clair que $d = -\mathcal{P}\nabla D(x^k, \lambda^k, \mu^k)$ est une direction de descente pour le Lagrangien augmenté en norme ℓ_1. Dans le cas contraire, il existe un vecteur $\bar{\lambda}$ tel que :

$$\nabla D(x^k, \lambda^k, \mu^k) = \sum_{j \in A_{\epsilon^k}} \bar{\lambda}_j^k \nabla c_j(x^k), \tag{5.8}$$

et il est, alors, possible de définir, pour une contrainte d'indice $t \in A_{\epsilon^k}$, le projecteur orthogonal $\mathcal{P}_t$ sur le noyau de $\{\nabla c_j(x^k)\}_{j \in A_{\epsilon^k} \setminus \{t\}}$. La variation $\nabla D^\top d_t$ le long de la direction $d_t = -\mathcal{P}_t \nabla D(x^k, \lambda^k, \mu^k) = -\bar{\lambda}_t^k \mathcal{P}_t \nabla c_t(x^k)$ est donnée par :

$$\bar{\lambda}_t^k d_t^\top \nabla c_t(x^k) + \frac{s_{tt}^k}{\mu^k} \sigma_t^k d_t^\top \nabla c_t(x^k) = -\left[(\bar{\lambda}_t^k)^2 + \frac{s_{tt}}{\mu^k} \sigma_t^k \bar{\lambda}_t^k\right] \left\|\mathcal{P}_t \nabla c_t(x^k)\right\|^2. \tag{5.9}$$

La direction d_t est alors une direction de descente sauf si $\left[(\bar{\lambda}_t^k)^2 + \frac{s_{tt}}{\mu^k}\sigma_t^k\bar{\lambda}_t^k\right]$ est négatif. Donc d_t n'est pas une direction de descente si :

$$\frac{-s_{tt}^k}{\mu^k} \leq \bar{\lambda}_t^k \leq 0 \text{ lorsque } \sigma_t^k = 1, \qquad \text{ou} \qquad 0 \leq \bar{\lambda}_t^k \leq \frac{s_{tt}^k}{\mu^k} \text{ lorsque } \sigma_t^k = -1. \tag{5.10}$$

Il faut aussi définir une stratégie de mise à jour des multiplicateurs de Lagrange pour l'algorithme. De manière similaire à la mise à jour des multiplicateurs dans le chapitre précédent, nous proposons la stratégie suivante :

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k + \bar{\lambda}_j^k & , \quad \text{si } j \in A_{\epsilon^k} \\ \lambda_j^k - \frac{s_{jj}^k}{\mu^k}\sigma_j^k & , \quad \text{si } j \in V_{\epsilon^k} \end{cases} \tag{5.11}$$

et nous notons :

$$\lambda^{k+1} = \hat{\lambda}(x^k, \lambda^k, \mu^k, S^k, \bar{\lambda}^k). \tag{5.12}$$

Si $\left\|c(x^k)\right\|$ est suffisamment petit pour que $V_{\epsilon^k} = \emptyset$, alors nous avons :

$$\nabla L(x^k, \lambda^{k+1}) = \nabla f(x^k) - \sum_{j=1}^{m} \lambda_j^k \nabla c_j(x^k) - \sum_{j \in A_{\epsilon^k}} \bar{\lambda}_j^k \nabla c_j(x^k) = 0, \tag{5.13}$$

où L est le Lagrangien du problème (5.1) :

$$L(x, \lambda) = f(x) - \sum_{j=1}^{m} \lambda_j c_j(x). \tag{5.14}$$

Pour la méthode classique du Lagrangien augmenté, la condition critique pour sélectionner le nouvel itéré est donnée par le gradient du Lagrangien augmenté qui coïncide avec le gradient du Lagrangien en utilisant la mise à jour des multiplicateurs. Pour la nouvelle méthode du Lagrangien augmenté en norme ℓ_1 il faut définir :

$$\begin{aligned} L_D(x, \lambda^k, \mu^k, \bar{\lambda}^k) &= D(x, \lambda^k, \mu^k) - \sum_{j \in A_{\epsilon^k}} \bar{\lambda}_j^k c_j(x) \\ &= f(x) - \sum_{j=1}^{m} \lambda_j^k c_j(x) + \tfrac{1}{\mu^k} \sum_{j \in V_{\epsilon^k}} s_{jj}\sigma_j^k c_j(x) - \sum_{j \in A_{\epsilon^k}} \bar{\lambda}_j^k c_j(x) \\ &= f(x) - \sum_{j \in V_{\epsilon^k}} (\lambda_j^k - \tfrac{s_{jj}}{\mu^k}\sigma_j^k c_j(x)) - \sum_{j \in A_{\epsilon^k}} (\lambda_j^k + \bar{\lambda}_j^k) c_j(x) \\ &= f(x) - \sum_{j=1}^{m} \lambda_j^{k+1} c_j(x) \\ &= L(x, \lambda^{k+1}). \end{aligned} \tag{5.15}$$

On peut alors définir la notion de criticalité pour l'algorithme du Lagrangien augmenté en norme ℓ_1. La condition critique permet de contrôler la qualité du prochain itéré x^k de l'algorithme.

Définition 1. *Soit $\omega \in \mathbb{R}^+$. Un point x est ω-critique à l'itération k si $\left\|\nabla L_D(x, \lambda^k, \mu^k, \bar{\lambda}^k)\right\| \leq \omega$ et si, pour tout $j \in A_{\epsilon^k}$, $\frac{-s^k_{jj}}{\mu^k} \leq \bar{\lambda}^k_j \leq 0$ si $\sigma^k_j = 1$, ou $0 \leq \bar{\lambda}^k_j \leq \frac{s^k_{jj}}{\mu^k}$ si $\sigma^k_j = -1$.*

L'algorithme du Lagrangien augmenté en norme ℓ_1 produit, à chaque itération k, une point x^{k+1} qui est ω^k-critique. Suivant les valeurs des contraintes en ce nouvel itéré x^{k+1}, l'algorithme met à jour les multiplicateurs λ^k ou diminue la valeur du terme de penalité μ^k. L'algorithme 13 décrit la boucle externe de la méthode du Lagrangien augmenté en norme ℓ_1.

5.2 La méthode de Coleman et Conn

La méthode de Coleman et Conn a été décrite en détail dans le chapitre précédent comme l'itération interne de l'algorithme de la fonction de pénalité exacte en norme ℓ_1. Cette approche ainsi que ses résultats de convergence sont présentés dans [42, 43]. Dans cette section, nous résumons les principaux résultats de convergence de cette méthode.

Dans le chapitre précédent, nous avons montré que la méthode de Coleman et Conn (algorithme 6) évalue plusieurs types de directions de descente. La définition de ces directions est basée sur une approximation de la matrice hessienne du problème (5.2) notée B^k et sur une matrice $Z^k \in \mathbb{R}^{n\times(n-t^k)}$, où $t^k = card(A_{\epsilon^k})$, qu'on trouve en résolvant le système suivant :

$$\left\{ \begin{array}{lcl} J_{A_{\epsilon^k}}(x^k)^\top Z^k & = & 0 \\ (Z^k)^\top Z^k & = & I_{n-t^k} \end{array} \right. \tag{5.16}$$

où $J_{A_{\epsilon^k}}(x^k)$ représente la matrice jacobienne des contraintes ϵ^k-actives . Suivant les cas, soit l'algorithme entreprend une recherche linéaire suivant une direction de descente h^k, soit l'itéré est mis à jour avec un pas égal à 1 dans la direction $h^k + v^k$ (voir la section 4.1 pour les détails de ces directions). Le théorème 1 de [43] stipule que l'algorithme 6 converge vers un point stationnaire de (5.2) et que, après un nombre fini d'itérations, seule la direction $h^k + v^k$ est choisie par l'algorithme.

Ce théorème permet de simplifier l'algorithme de Coleman et Conn quand l'itéré courant est proche d'une solution x^* et que la seule direction de descente considérée et h^k+v^k. Donc, dans le voisinage d'une solution x^*, l'approche de Coleman et Conn se réduit à l'algorithme local 14.

Une suite $\{u_n\}$ a une convergence superlinéaire en deux étapes vers u^* si :

$$\lim_{n\to\infty} \frac{\|u_{n+1} - u^*\|}{\|u_{n-1} - u^*\|} = 0. \tag{5.17}$$

Algorithme 13: Lagrangien augmenté en norme ℓ_1 (boucle externe)

1-Initialisations :

μ_0, η_0, ω_0, $\tau < 1$, $\gamma_1 < 1$, $\omega_* \ll 1$, $\eta_* < 1$, $\alpha_\omega < 1$, $\beta_\omega < 1$, $\alpha_\eta < 1$, $\beta_\eta < 1$

Deux matrices diagonales S_1 et S_2 telles que $0 < diag(S_1^{-1}) \leq diag(S_2) < \infty$ (comparaison des diagonales élément par élément)

$k \leftarrow 0$, λ^k, $\mu^k \leftarrow \mu_0$, $\alpha^k \leftarrow \min\{\mu^k, \gamma_1\}$, $\eta^k \leftarrow \eta_0(\alpha^k)^{\alpha_\eta}$, $\omega^k \leftarrow \omega_0(\alpha^k)^{\alpha_\omega}$

2-Itération interne :

Définir une matrice diagonale S^k telle que $diag(S_1^{-1}) \leq diag(S^k) \leq diag(S_2)$

Trouver $x^{k+1} \in \mathcal{X}$ tel que x^{k+1} est ω^k-critique :

$$\left\|\nabla L_D(x^{k+1}, \lambda^k, \mu^k, \bar{\lambda}^k)\right\| \leq \omega^k$$

3-Conditions d'arrêt :

Si $\left\|\nabla L_D(x^{k+1}, \lambda^k, \mu^k, \bar{\lambda}^k)\right\| \leq \omega_*$ et $\left\|c(x^{k+1})\right\| \leq \eta_*$

STOP.

Sinon Si $\left\|c(x^{k+1})\right\| \leq \eta^k$

Aller à **4**

Sinon

Aller à **5**

fin Si

4-Mise-à-jour des multiplicateurs de Lagrange :

$$\begin{aligned}
\lambda^{k+1} &\leftarrow \begin{cases} \lambda_j^k + \bar{\lambda}_j^k, & \text{si } j \in A_\epsilon \\ \lambda_j^k - \frac{\sigma_j^k}{\mu^k}, & \text{si } j \in V_\epsilon \end{cases} \\
\mu^{k+1} &\leftarrow \mu^k \\
\alpha^{k+1} &\leftarrow \min\{\mu^{k+1}, \gamma_1\} \\
\eta^{k+1} &\leftarrow \eta^k(\alpha^{k+1})^{\beta_\eta} \\
\omega^{k+1} &\leftarrow \omega^k(\alpha^{k+1})^{\beta_\omega}
\end{aligned}$$

$k \leftarrow k + 1$

Aller à **2**

5-Réduction du paramètre de pénalité :

$$\begin{aligned}
\lambda^{k+1} &\leftarrow \lambda^k \\
\mu^{k+1} &\leftarrow \tau\mu^k \\
\alpha^{k+1} &\leftarrow \min\{\mu^{k+1}, \gamma_1\} \\
\eta^{k+1} &\leftarrow \eta_0(\alpha^{k+1})^{\alpha_\eta} \\
\omega^{k+1} &\leftarrow \omega_0(\alpha^{k+1})^{\alpha_\omega}
\end{aligned}$$

$k \leftarrow k + 1$

Aller à **2**

Le théorème suivant permet d'établir que l'algorithme 14 converge superlinéairement en deux étapes.

Théorème 1. *En supposant que :*

(i) Les fonctions f et c_j, $j \in \{1, 2, \ldots, m\}$, sont deux fois continûment différentiables,

(ii) La suite des points $\{x^k\}$ générée par l'algorithme 14 converge vers x^*,

Algorithme 14: L'approche de Coleman et Conn simplifiée

1: Initialisations : $k \leftarrow 1$ et x^0 suffisamment proche de x^*
2: Évaluer $\bar{\lambda}^k$
3: Mettre à jour $(Z^k)^\top B^k Z^k$ (en la gardant définie positive)
4: Évaluer les directions de descente h^k et v^k
5: Mettre à jour $x^{k+1} \leftarrow x^k + h^k + v^k$

(iii) Les colonnes de la matrice Jacobienne des contraintes ϵ-actives $\hat{A}(x^*)$ sont linéairement indépendantes,

(iv) Les conditions suffisantes de second ordre sont satisfaites en x^* (voir dans [62]),

(v) Il existe deux scalaires b_1 et b_2 tels que $0 < b_1 \leq b_2$ et pour tout $y \in \mathbb{R}^{n-t}$ et toute itération k :

$$b_1 \|y\|^2 \leq y^\top (Z^{k\top} B^k Z^k) y \leq b_2 \|y\|^2 , \tag{5.18}$$

(vi) $Z^{k\top} B^k Z^k$ converge vers $Z^{*\top}(\nabla^2 f(x^*) - \sum_{j=1}^{t} \bar{\lambda}_j^* \nabla^2 c_j(x^*)) Z^*$,

Alors :

$$\lim_{k\to\infty} \frac{\left\|x^{k+1} - x^*\right\|}{\|x^{k-1} - x^*\|} = 0, \tag{5.19}$$

Un algorithme est dit localement convergent si, à partir d'un point assez proche d'une solution stationnaire, la suite des itérés générée par la méthode converge vers ce point stationnaire. Au contraire, la convergence globale assure qu'un algorithme converge vers une solution stationnaire, peu importe le point de départ choisi. Le théorème suivant permet d'établir la convergence locale de l'algorithme 14 :

Théorème 2. *Pour la suite de points $\{x^k\}$ générée par l'algorithme 14, si x^0 et $Z^{0\top} B^0 Z^0$ sont suffisamment proche de x^* et $Z^{*\top}(\nabla^2 f(x^*) - \sum_{j=1}^{t} \bar{\lambda}_j^* \nabla^2 c_j(x^*)) Z^*$, respectivement, alors :*

$$\lim_{k\to\infty} x^k = x^*. \tag{5.20}$$

Ce dernier résultat, couplé avec le théorème 1 de [43], permet d'établir la convergence globale de la méthode de Coleman et Conn. Le taux de convergence superlinéaire s'étend également à la méthode de Coleman et Conn complète, puisque celle ci se réduit automatiquement à l'algorithme 14. Un autre résultat important de cette méthode réside dans le fait que le paramètre de pénalité μ^k converge loin de zéro si x^0 satisfait des conditions suffisantes de second ordre [62]. C'est à dire qu'il existe un seuil μ tel que $\mu^k > \mu$ pour tout k.

5.3 Analyse de convergence du Lagrangien augmenté en norme ℓ_1

La section suivante est inspirée de l'article [44] qui développe les résultats de convergence pour la méthode du Lagrangien augmenté. Les résultats de convergence avancés pour la nouvelle méthode suivent le même enchaînement de [44] avec une adaptation des preuves pour rester conforme avec la norme ℓ_1.

5.3.1 Analyse de convergence globale

Dans la suite de ce chapitre, nous allons utiliser les hypothèses suivantes :

Hypothèse 1. *Les fonctions f et $\{c_j\}_{j\in\{1,2,\ldots,m\}}$ sont deux fois continûment différentiables pour tout $x \in \mathbb{R}^n$.*

Hypothèse 2. *Les itérés $\{x^k\}$ appartiennent à un ensemble fermé borné Ω.*

Hypothèse 3. *Les colonnes de la matrice Jacobienne des contraintes actives $\hat{A}(x^*)$ sont linéairement indépendantes en tout point x^* vers lequel la suite $\{x^k\}$ converge.*

Nous commençons par analyser la convergence de l'algorithme 13 dans le cas où ω_* (le seuil sur $\|\nabla L_D\|$) et η_* (le seuil sur $\|c\|$) sont nuls. Le lemme suivant suggère que les estimateurs des multiplicateurs de Lagrange sont adéquats.

Lemme 1. *Si μ^k converge vers 0 dans l'algorithme 13, alors $\mu^k \left\|\lambda^k\right\|$ converge aussi vers 0.*

Démonstration. La convergence de μ^k vers 0 signifie que l'étape 5 de l'algorithme est exécutée une infinité de fois. Soit $K = \{k_0, k_1, k_2, \ldots\}$ l'ensemble des itérations où l'étape 5 de l'algorithme est exécutée et pour lesquelles :

$$\mu^k \leq \min\left\{\gamma_1, \left(\frac{1}{2}\right)^{\frac{1}{\beta^\eta}}\right\}. \tag{5.21}$$

Entre deux itérations successives k_i et k_{i+1} de K, c'est l'étape 4 de l'algorithme qui est exécutée et pour l'itération $k_i + l$ où $k_i < k_i + l \leq k_{i+1}$ nous avons :

$$\lambda^{k_i+l} = \lambda^{k_i} + \sum_{r=1}^{l-1} S^{k_i+r}\varsigma^{k_i+r}, \tag{5.22}$$

avec :

$$\xi^{k_i+r} = \begin{cases} \bar{\lambda}_j^{k_i+r} & , \quad \text{si } j \in A_\epsilon \\ -\frac{\sigma_j^{k_i+r}}{\mu^{k_i+r}} & , \quad \text{si } j \in V_\epsilon \end{cases} \tag{5.23}$$

et

$$\mu^{k_{i+1}} = \mu^{k_{i+1}-1} = \cdots = \mu^{k_i+r} = \cdots = \mu^{k_i+1} = \tau\mu^{k_i}. \tag{5.24}$$

La somme de (5.22) est nulle si $l = 1$. Comme l'étape 4 de l'algorithme est exécutée entre les itérations k_i et k_{i+1}, alors le test de l'étape 3 stipule que pour l'itération $k_i + l$, $l > 1$, nous avons :

$$\begin{array}{rcl} \left\| c(x^{k_i+l+1}) \right\| & \leq & \eta^{k_i+l} \\ & \leq & \eta_0(\mu^{k_i+1})^{\alpha_\eta+\beta_\eta(l-1)}. \end{array} \tag{5.25}$$

Donc, l'ensemble des contraintes ϵ-violées devient éventuellement vide. Soit l'ensemble $K' = \{k'_0, k'_1, k'_2, \ldots\}$ tel que $K' \subset K$ et $V_\epsilon = \emptyset$. À l'itération $k'_i + l$, l'équation (5.22) devient :

$$\lambda^{k'_i+l} = \lambda^{k'_i} + \sum_{r=1}^{l-1} S^{k'_i+r}\bar{\lambda}^{k'_i+r}. \tag{5.26}$$

Nous pouvons évaluer $\bar{\lambda}^{k_i+r}$ via :

$$\begin{array}{rcl} \bar{\lambda}^{k'_i+r} & = & \hat{A}(x^{k'_i+r})^\top(\hat{A}(x^{k'_i+r})\hat{A}(x^{k'_i+r})^\top)^{-1}\nabla L(x^{k'_i+r}, \lambda^{k'_i+r}) \\ & = & \hat{A}(x^{k'_i+r})^\top(\hat{A}(x^{k'_i+r})\hat{A}(x^{k'_i+r})^\top)^{-1}\nabla L_D(x^{k'_i+r}, \lambda^{k'_i+r-1}, \mu^{k'_i+r-1}, \bar{\lambda}^{k'_i+r-1}). \end{array} \tag{5.27}$$

D'après les hypothèses (1) et (3), pour des valeurs suffisamment grandes de k, il existe une constante a_1 telle que :

$$\left\| \hat{A}(x^{k'_i+r})^\top(\hat{A}(x^{k'_i+r})\hat{A}(x^{k'_i+r})^\top)^{-1} \right\| \leq a_1. \tag{5.28}$$

En utilisant cette équation et le fait que $x^{k'_i+r-1}$ est $\omega^{k'_i+r-1}$-critique (selon l'itération interne 2 de l'algorithme 13), nous avons :

$$\left\| \bar{\lambda}^{k'_i+r} \right\| \leq a_1\omega^{k'_i+r-1}, \tag{5.29}$$

et en combinant (5.26), (5.29) et la borne supérieure $s_2 = \|S_2\|$ de S^k, nous trouvons :

$$\begin{aligned}
\left\|\lambda^{k'_i+l}\right\| &\leq \left\|\lambda^{k'_i}\right\| + \sum_{r=1}^{l-1} s_2 a_1 \omega^{k'_i+r-1} \\
&\leq \left\|\lambda^{k'_i}\right\| + a_1 s_2(\omega_0 + \sum_{r=2}^{l-1} \omega^{k'_i+r-1}) \\
&\leq \left\|\lambda^{k'_i}\right\| + a_1 s_2 \omega_0 (1 + (\mu^{k'_{i+1}})^{\alpha_\omega} \sum_{r=2}^{l-1} (\mu^{k'_{i+1}})^{\beta_\omega(r-2)}) \\
&\leq \left\|\lambda^{k'_i}\right\| + a_1 s_2 \omega_0 (1 + (\mu^{k'_{i+1}})^{\alpha_\omega} \frac{1}{1-(\mu^{k'_{i+1}})^{\beta_\omega}}) \\
&\leq \left\|\lambda^{k'_i}\right\| + a_1 s_2 \omega_0 (1 + 2(\mu^{k'_{i+1}})^{\alpha_\omega})
\end{aligned} \tag{5.30}$$

et nous obtenons :

$$\begin{aligned}
\mu^{k'_i+l}\left\|\lambda^{k'_i+l}\right\| &\leq \tau\mu^{k'_i}\left\|\lambda^{k'_i}\right\| + a_1 s_2 \omega_0 \mu^{k'_i+l}(1 + 2(\mu^{k'_{i+1}})^{\alpha_\omega}) \\
&\leq \tau\mu^{k'_i}\left\|\lambda^{k'_i}\right\| + a_1 s_2 \omega_0 (1 + 2\mu_0^{\alpha_\omega})\mu^{k'_i+l}.
\end{aligned} \tag{5.31}$$

Cette inégalité est satisfaite pour $l = 1$. À partir de (5.31) nous avons :

$$\mu^{k'_{i+1}}\left\|\lambda^{k'_{i+1}}\right\| \leq \tau\mu^{k'_i}\left\|\lambda^{k'_i}\right\| + a_1 s_2 \omega_0 (1 + 2\mu_0^{\alpha_\omega})\mu^{k'_{i+1}}. \tag{5.32}$$

La séquence $\mu^{k'_i}\left\|\lambda^{k'_i}\right\|$ converge vers 0. En posant $\alpha_i = \mu^{k'_i}\left\|\lambda^{k'_i}\right\|$ et $\beta_i = a_1 s_2 \omega_0 (1 + 2\mu_0^{\alpha_\omega})\mu^{k'_i}$, nous pouvons réécrire (5.31) en :

$$\alpha_{i+1} \leq \tau\alpha_i + \tau\beta_i \text{ et } \beta_i = \tau^i \beta_0. \tag{5.33}$$

Par induction, nous trouvons :

$$\begin{aligned}
\alpha_i &\leq \tau\alpha_{i-1} + \tau\beta_{i-1} \\
&\leq \tau^i\alpha_0 + \sum_{r=1}^{i} \tau^r \beta_{i-r} \\
&\leq \tau^i\alpha_0 + \sum_{r=1}^{i} \tau^r \tau^{i-r} \beta_0 \\
&\leq \tau^i\alpha_0 + i\tau^i\beta_0.
\end{aligned} \tag{5.34}$$

Cette inégalité montre que α_i converge vers 0 avec l'augmentation de i. Étant donné que β_i converge aussi vers 0, alors la partie droite de (5.32) tend vers 0 et le lemme est établi. ■

Par la suite, nous considérons que $\mathcal{X} = \{x \in \mathbb{R}^n : x \geq 0\}$ et nous définissons la projection $\mathcal{P}_\mathcal{X}$ élément par élément :

$$(\mathcal{P}_{\mathcal{X}}(x))_i = \begin{cases} 0 & \text{si } x_i \leq 0, \\ x_i & \text{sinon .} \end{cases} \tag{5.35}$$

Nous utiliserons aussi la projection suivante :

$$\mathcal{P}_{\mathcal{X}}(x, v) = x - \mathcal{P}_{\mathcal{X}}(x - v). \tag{5.36}$$

Pour chaque composante x_i^k de x^k nous avons deux possibilités :

$$\text{Cas 1} : 0 \leq x_i^k \leq (\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))_i. \tag{5.37}$$

$$\text{Cas 2} : (\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))_i < x_i^k. \tag{5.38}$$

Pour le premier cas (5.37), x_i^k est dite variable dominée et nous avons :

$$(\mathcal{P}_{\mathcal{X}}(x^k, \nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)))_i = x_i^k. \tag{5.39}$$

Pour le second cas (5.38), x_i^k est dite variable flottante et satisfait :

$$(\mathcal{P}_{\mathcal{X}}(x^k, \nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)))_i = (\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))_i. \tag{5.40}$$

L'algorithme développé construit une suite d'itérés qui force $\mathcal{P}_{\mathcal{X}}(x^k, \nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))$ à tendre vers 0. Les variables dominées sont alors poussées vers 0 alors que les variables flottantes restent libres. De plus, s'il existe une suite convergente $\{x^k\}_{k \in K}$ qui converge vers une limite x^*, nous décomposons l'ensemble $\mathcal{N} = \{1, 2, \ldots, n\}$ en quatre sous-ensembles disjoints :

$$I_1 = \{i \in \{1, 2, \ldots, n\} : x_i^k \text{ est flottante pour } k \in K \text{ assez grand et } x_i^* > 0\}, \tag{5.41}$$

$$I_2 = \{i \in \{1, 2, \ldots, n\} : x_i^k \text{ est dominée pour } k \in K \text{ assez grand}\}, \tag{5.42}$$

$$I_3 = \{i \in \{1, 2, \ldots, n\} : x_i^k \text{ est flottante pour } k \in K \text{ assez grand et } x_i^* = 0\}, \tag{5.43}$$

$$I_4 = \mathcal{N} \setminus (I_1 \cup I_2 \cup I_3). \tag{5.44}$$

Par abus de langage, nous écrivons que x_i appartient à un ensemble I_k (pour $k \in \{1, 2, 3, 4\}$) alors qu'il faut plutôt considérer que c'est l'indice i qui appartient à I_k. De même, quand nous parlons de composantes d'un vecteur donné dans l'ensemble I_k, nous voulons dire les indices des composantes de ce vecteur dans l'ensemble I_k. Le lemme suivant présente des propriétés importantes des

éléments de ces différents ensembles.

Lemme 2. *Supposons que $\mathcal{P}_{\mathcal{X}}(x^k, \nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))$ converge vers 0 avec l'augmentation de k et que la sous-suite $\{x^k\}_{k\in K}$ converge vers une limite x^*. Alors :*

(i) Les variables dans les ensembles I_2, I_3 et I_4 convergent vers leurs bornes (dans ce cas, 0),

(ii) Les composantes de $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ dans les ensembles I_1 et I_3 tendent vers 0,

(iii) Si une composante de $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ dans l'ensemble I_4 converge vers une limite finie, alors cette limite est 0.

Démonstration. (i) Le résultat est vrai pour les variables de I_2 d'après (5.39), pour les variables de I_3 par définition. Pour les variables I_4, il existe une sous-suite convergente $\{x_i^k\}_k$ de variables dominées qui converge vers 0 d'après (5.39).

(ii) Les composantes de $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ dans les ensembles I_1 et I_3 tendent vers 0 d'après l'équation (5.40).

(iii) Pour les variables I_4, il existe une sous-suite convergente $\{x_i^k\}_k$ de variables flottantes, pour qui $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ converge vers 0 d'après (5.40).

■

Il est parfois pratique de regrouper les variables de I_3 et I_4 ensemble dans $I_5 = I_3 \cup I_4$. L'ensemble I_5 contient les variables qui sont égales à 0 dans la solution du problème et qui peuvent correspondre aux composantes nulles de $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$. Ces variables sont potentiellement dégénérées à la solution du problème non linéaire.

Pour le vecteur $\nabla f(x)$, $\hat{\nabla} f(x)$ indique les composantes de $\nabla f(x)$ indexées par I_1. De même, pour une matrice M, $\hat{M}$ représente les colonnes de M indexées par I_1. Par conséquent, nous redéfinissons $\hat{A}$ dans l'hypothèse (3) comme étant la matrice groupant les colonnes de la matrice Jacobienne correspondant à l'ensemble I_1. Nous définissons également l'estimation des multiplicateurs de Lagrange au sens des moindres carrées (correspondant à l'ensemble I_1) :

$$\lambda(x) = \left(\left(\hat{A}(x)\right)^+\right)^\top \hat{\nabla} f(x), \tag{5.45}$$

où l'inverse généralisée $\left(\hat{A}(x)\right)^+$ est donnée par :

$$\left(\hat{A}(x)\right)^+ = \hat{A}(x)^\top \left(\hat{A}(x)\hat{A}(x)^\top\right)^{-1}. \tag{5.46}$$

Lemme 3. *Considérons que l'hypothèse* (1) *est satisfaite et soit* $\{x^k\}_{k\in K} \in \mathcal{X}$ *une sous-suite satisfaisant l'hypothèse* (2) *et qui converge vers un point* x^* *pour lequel l'hypothèse 3 est satisfaite. Soit* $\lambda^* = \bar{\lambda}(x^*)$*, où* $\bar{\lambda}$ *est donnée par :*

$$\nabla D(x, \lambda^k, \mu^k) = \sum_{j\in A_\epsilon} \bar{\lambda}_j^k \nabla c_j(x) \tag{5.47}$$

et soit $J_{A_\epsilon}(x)$ *la matrice jacobienne des contraintes* ϵ*-actives. Supposons que* $\{\lambda^k\}_{k\in K}$ *est une suite quelconque de vecteurs, que* $\{S^k\}_{k\in K}$ *est une suite quelconque de matrices diagonales telle que* $0 < S_1^{-1} \leq S^k \leq S_2 < \infty$ *et* $\{\mu^k\}_{k\in K}$ *une suite non croissante de scalaires positifs. De plus supposons que :*

$$\left\|\mathcal{P}_{\mathcal{X}}(x^k, \nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))\right\| \leq \omega^k \tag{5.48}$$

avec ω^k *une suite de scalaires positifs qui converge vers* 0 *avec l'augmentation de* $k \in K$ *et pour tout* $j \in A_\epsilon$*, soit* $\frac{-s_{jj}^k}{\mu^k} \leq \bar{\lambda}_j^k \leq 0$ *si* $\sigma_j^k = 1,$ *soit* $0 \leq \bar{\lambda}_j^k \leq \frac{s_{jj}^k}{\mu^k}$ *si* $\sigma_j^k = -1$*. Alors :*

(i) Il existe des constantes positives a_1*,* a_2*,* s_1 *et un entier* k_0 *tels que :*

$$\left\|\hat{\lambda}(x^k, \lambda^k, \mu^k, S^k, \bar{\lambda}^k) - \lambda^*\right\| \leq a_1\omega^k + a_2\left\|x^k - x^*\right\| \tag{5.49}$$

et

$$\left\|\lambda(x^k) - \lambda^*\right\| \leq a_1\omega^k + a_2\left\|x^k - x^*\right\| \tag{5.50}$$

pour tout $k \geq k_0$ *(*$k \in K$*).*
Supposons en plus que $c(x^*) = 0$*. Alors :*

(ii) Le point x^* *est un point Karush-Kuhn-Tucker pour le problème* (5.1) *et les suites* $\{\lambda(x^k)\}_{k\in K}$ *et* $\{\bar{\lambda}(x^k, \lambda^k, \mu^k, S^k, \bar{\lambda}^k)\}_{k\in K}$ *convergent vers le vecteur des multiplicateurs de Lagrange* λ^**.*

(iii) Le gradient $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ *converge de telle façon que l'équation 5.13 est satisfaite pour tout* $k \in K$ *et* $\nabla L(x^*, \lambda^*) = 0$*.*

Démonstration. Nous avons vu précédemment que grâce aux hypothèses (1) et (3), nous avons pour $k \in K$ suffisamment grand, $\hat{A}(x^k)^+$ est borné et converge vers la matrice $\left(\hat{A}(x^*)\right)^+$. Il existe donc une constante $a_1 > 0$:

$$\left\|\left(\hat{A}(x^k)\right)^+\right\| \leq a_1. \tag{5.51}$$

Comme les variables de I_1 sont flottantes, les équations (5.47), (5.12), (5.40) couplées avec le

critère d'arrêt (5.48) et l'égalité (5.15) donnent :

$$\left\|\hat{\nabla}L(x^k,\lambda^{k+1})\right\| = \left\|\hat{\nabla}f(x^k) - \hat{A}(x)^\top\hat{\lambda}(x^k)\right\| \leq \omega^k. \tag{5.52}$$

Par hypothèse, $\lambda(x)$, defini dans (5.45), est borné pour tout x dans le voisinage de x^*. Grâce à $\bar{\lambda}^k$ comme estimation au sens des moindres carrées de $\nabla D(x,\lambda^k,\mu^k) = \sum_{j\in A_\epsilon}\hat{\lambda}_j^k\nabla c_j(x)$ et aux équations (5.51) et (5.52), il est possible de montrer que :

$$\begin{aligned}
\left\|\lambda(x^k) - \hat{\lambda}^k\right\| &= \left\|\left(\left(\hat{A}(x^k)\right)^+\right)^\top\hat{\nabla}f(x^k) - \hat{\lambda}^k\right\| \\
&= \left\|\left(\left(\hat{A}(x^k)\right)^+\right)^\top\left[\hat{\nabla}f(x^k) - \hat{A}(x^k)^\top\hat{\lambda}(x^k)\right]\right\| \\
&\leq \left\|\left(\hat{A}(x^k)\right)^+\right\|\,\omega^k \\
&\leq a_1\,\omega^k.
\end{aligned} \tag{5.53}$$

Puisque $\lambda(x)$ est différentiable, alors le lemme 2.2 de l'article [44] démontre que :

$$\nabla\lambda(x) = -(\left(\hat{A}(x)\right)^+)^\top\hat{H}_L(x,\lambda(x)) - \left(\hat{A}(x)\hat{A}(x)^\top\right)^{-1}R(x), \tag{5.54}$$

où la $i^{ème}$ ligne de $R(x)$ est $(\hat{\nabla}f(x) + \hat{A}(x)^\top\lambda(x))^\top\hat{H}_i(x)$, H est la matrice hessienne de f, H_i est la matrice hessienne de c_i et $H_L(x,\lambda)$ est la matrice hessienne du Lagrangien du problème. Par l'intermédiaire du théorème des accroissements finis et de l'équation 5.54 nous avons :

$$\lambda(x^k) - \lambda^* = \int_0^1 \nabla\lambda(x(s))ds\cdot(x^k - x^*), \tag{5.55}$$

où $x(s) = x^k + s(x^* - x^k)$. Le terme à l'intérieur de l'intégrale est borné pour tout x suffisamment proche de x^*, donc il existe une constante $a_2 > 0$, pour k suffisamment grand, telle que :

$$\left\|\lambda(x^k) - \lambda^*\right\| \leq a_2\left\|x^k - x^*\right\|. \tag{5.56}$$

Donc $\lambda(x^k)$ converge vers λ^*. En utilisant (5.53) et (5.56), il est possible de démontrer que :

$$\left\|\hat{\lambda}^k - \lambda^*\right\| \leq \|\hat{\lambda}^k - \lambda(x^k)\| + \left\|\lambda(x^k) - \lambda^*\right\| \leq a_1\,\omega^k + a_2\left\|x^k - x^*\right\|. \tag{5.57}$$

Comme ω^k converge vers zéro avec l'augmentation de k, (5.49) permet de montrer que $\hat{\lambda}^k$ converge vers λ^*. D'après l'équation (5.52) nous avons :

$$\nabla L(x^*, \lambda^*) = \nabla f(x^*) - \hat{A}(x^*)^\top \lambda^* = 0. \tag{5.58}$$

Supposons que $c(x^*) = 0$. D'après (5.15), le gradient $\nabla L_D^k(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ converge vers $\nabla L(x^*, \lambda^*)$. Le lemme (2) et la convergence de $\nabla\ L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ vers $\nabla L(x^*, \lambda^*)$ montrent que la condition des écarts complémentaires :

$$\nabla L(x^*, \lambda^*) x^* = 0 \tag{5.59}$$

est satisfaite. Les variables de l'ensemble I_1 sont positives en x^*. Les composantes de $\nabla L(x^*, \lambda^*)$ indexées par I_2 sont toutes positives ou égales à zéro d'après (5.37). On a alors les conditions suivantes :

$$\begin{array}{lll} x_i^* > 0 \quad \text{et} \quad (\nabla L(x^*, \lambda^*))_i = 0 \quad \text{pour } i \in I_1, \\ x_i^* = 0 \quad \text{et} \quad (\nabla L(x^*, \lambda^*))_i \geq 0 \quad \text{pour } i \in I_2, \\ x_i^* = 0 \quad \text{et} \quad (\nabla L(x^*, \lambda^*))_i = 0 \quad \text{pour } i \in I_5. \end{array} \tag{5.60}$$

Comme $c(x^*) = 0$, les conditions (5.60) montrent que x^* est un point de Karush-Kuhn-Tucker et λ^* correspond au vecteur des multiplicateurs de Lagrange. De plus, (5.49) et (5.50) assurent la convergence de $\hat{\lambda}(x^k, \lambda^k,\ S^k,\ \mu^k, \bar{\lambda}^k)$ et $\lambda(x^k)$ vers λ^* pour $k \in K$. ■

Le théorème suivant établit la convergence globale de l'algorithme 13.

Théorème 3. *Considérons que l'hypothèse 1 est satisfaite et soit x^* le point limite d'une suite $\{x^k\}$ générée par l'algorithme 13 et pour lesquels les hypothèses* (2) *et* (3) *sont satisfaites. De plus, soit K un ensemble infini d'indices de la sous-suite $\{x^k\}$ dont x^* est la limite. Les conclusions (i), (ii) et (iii) du lemme 3 sont alors valides.*

Démonstration. Les hypothèses du théorème permettent d'établir le résultat (i) du lemme 3. Pour les résultats (ii) et (iii) du lemme 3, il suffit de montrer que $c(x^*) = 0$. La méthode de Coleman et Conn assure que le terme de pénalité μ^k est borné loin de zéro. Par conséquent, l'étape 4 de l'algorithme est exécutée à chaque itération pour k suffisamment grand. Ceci veut dire que le test de l'étape 3 $\left\|c(x^k)\right\| \leq \eta^k$ est toujours satisfait pour k suffisamment grand et puisque η^k tend vers zéro alors $c(x^k)$ converge vers zéro. Il s'ensuit que les résultats (ii) et (iii) sont établis. ■

Le théorème 3 reste valable pour n'importe quelles suites ω^k et η^k convergentes vers zéro.

5.3.2 Analyse de convergence asymptotique

Pour la suite de cette analyse de convergence, nous notons $\mathcal{H}_L(x^*, \lambda^*)_{[\mathcal{J}_1,\mathcal{J}_2]}$ la matrice formée en sélectionnant les lignes et les colonnes de $\mathcal{H}_L(x^*, \lambda^*)$ indexées par $\mathcal{J}_1$ et $\mathcal{J}_2$ et $\mathcal{A}(x^*)_{[J_1]}$ la matrice issue de $\mathcal{A}(x^*)$ en choisissant uniquement les colonnes indéxées par $\mathcal{J}_1$ et nous introduisons les deux hypothèses suivantes :

Hypothèse 4. *Les dérivées secondes des fonctions f et c_j sont Lipschitz en tout point de $\mathcal{X}$.*

Hypothèse 5. *Supposons que (x^*, λ^*) est un point de Karush-Kuhn-Tucker pour le problème* (5.1) *et définissons :*

$$\begin{array}{rcl} \mathcal{J}_1 & = & \{i \in \{1, 2, \ldots, n\} : x_i^* > 0 \quad \textit{et} \quad (\nabla L(x^*, \lambda^*))_i = 0\}, \\ \mathcal{J}_2 & = & \{i \in \{1, 2, \ldots, n\} : x_i^* = 0 \quad \textit{et} \quad (\nabla L(x^*, \lambda^*))_i = 0\}. \end{array} \tag{5.61}$$

De plus, nous supposons que la matrice

$$\begin{bmatrix} \mathcal{H}_L(x^*, \lambda^*)_{[\mathcal{J},\mathcal{J}]} & (\mathcal{A}(x^*)_{[J]})^\top \\ \mathcal{A}(x^*)_{[J]} & 0 \end{bmatrix}$$

est inversible pour tout ensemble $\mathcal{J}$, où $\mathcal{J}$ est tout ensemble formé par l'union de $\mathcal{J}_1$ et d'un sous-ensemble quelconque de $\mathcal{J}_2$.

Les inégalités (5.49) et (5.50) font intervenir le terme $\left\|x^k - x^*\right\|$; ceci pose un problème vu qu'on veut étudier le taux de convergence dans cette section. Le lemme suivant permet d'éliminer cette dépendance.

Lemme 4. *Considérons que l'hypothèse* (1) *est satisfaite. Soit $x^k \in \mathcal{X},\ k \in K$, une sous-suite qui converge vers le point KKT x^* et pour lequel les hypothèses* (1)*,* (4)*, et* (5) *sont satisfaites, et soit λ^* le vecteur des multiplicateurs de Lagrange. Supposons que $\{\lambda(x^k)\},\ k \in K$, est une suite de vecteurs, que $\{S^k\},\ k \in K$, est une séquence de matrices diagonales telle que $0 < S_1^{-1} \leq S^k \leq S_2 < \infty$, et que $\{\mu^k\},\ k \in K$, est une suite décroissante de scalaires positifs telle que $\mu^k \left\|\lambda^k - \lambda^*\right\|$ converge vers zéro avec l'augmentation de k. De plus, supposons que :*

$$\left\|\mathcal{P}_{\mathcal{X}}(x^k,\ \nabla_x\, L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k))\right\| \leq \omega^k, \tag{5.62}$$

avec ω^k des scalaires positifs qui convergent vers zéro avec l'augmentation de $k \in K$. Il existe des constantes positives, $\bar{\mu},\ a_3,\ a_4,\ a_5,\ a_6$, et s_1 et une valeur entière k_0 telle que, si $\mu^{k_0} \leq \bar{\mu}$ alors :

$$\left\|x^k - x^*\right\| \leq a_3\, \omega^k + a_4\, \eta^k, \tag{5.63}$$

$$\left\|\hat{\lambda}(x^k, \lambda^k,\ S^k,\ \mu^k,\ \bar{\lambda}^k) - \lambda^*\right\| \leq a_5\ \omega^k + a_{17}\ \eta^k, \tag{5.64}$$

$$\textit{et}\ \left\|c(x^k)\right\| \leq a_6\ \omega^k + a_{16}\ \eta^k, \tag{5.65}$$

pour tout $k \geq k_0$ *et* $k \in K$.

Démonstration. On note g_L et H_L respectivement le gradient et la matrice hessienne du Lagrangien du problème(5.1) par rapport à x. Au point (x^*, λ^*) nous utilisons la notation g_L^* et H_L^*.

Les hypothèses (4) et (5) et le fait que μ^k est borné loin de zéro montrent que pour k suffisamment grand nous avons :

$$\left\|c(x^{k+1})\right\| \leq \eta^k. \tag{5.66}$$

Nous commençons par analyser l'état des variables en approchant x^*. Nous choisissons un k assez grand pour que les ensembles I_1 et I_2 soient déterminés. Pour les variables restantes, celles dans I_3 sont flottantes et celles dans I_4 alternent entre flottantes et dominées pour $k \in K$. Soit $\bar{K} \subset K$ tel que :

(i) $I_5 = I_6 \cup I_7$ avec $I_6 \cap I_7 = \emptyset$,

(ii) Les variables de I_6 sont flottantes pour tout $k \in \bar{K}$,

(iii) Les variables de I_7 sont dominées pour tout $k \in \bar{K}$.

L'ensemble I_3 est contenu dans I_6. Le nombre de sous-ensembles de type $\bar{K}$ est fini et pour k suffisamment grand, k appartient forcément à un tel sous-ensemble. Donc, il suffit de prouver le lemme pour $k \in \bar{K}$. Pour $k \in \bar{K}$ nous définissons :

$$I_F = I_1 \cup I_6 \text{ et } I_D = I_2 \cup I_7. \tag{5.67}$$

Les variables de I_F sont flottantes alors que celles de I_D sont dominées. D'après l'approche de Coleman et Conn, l'inégalité (5.66) et l'hypothèse que x^k converge vers x^*, nous avons $c(x^*) = 0$. Donc, en utilisant le résultat (i) du lemme 3, nous avons, pour k suffisamment grand dans $\bar{K}$:

$$\left\|\hat{\lambda}^k - \lambda^*\right\| \leq a_1\ \omega^k + a_2 \left\|x^k - x^*\right\|, \tag{5.68}$$

avec λ^* le vecteur des multiplicateurs de Lagrange (selon le résultat (ii) du lemme 3). Donc, $\bar{\lambda}^k$ converge vers λ^* et, selon le lemme (3)(iii), $\nabla\ L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ converge vers g_L^*. En utilisant le lemme 2, nous avons :

$$x_i^* = 0 \text{ pour tout } i \in I_D \text{ et } (g_L^*)_i = 0 \text{ pour tout } i \in I_F. \tag{5.69}$$

En utilisant le développement de Taylor et l'équation (5.15) nous avons :

$$\begin{aligned} \nabla_x L_D^k(x^k, \lambda^k, \mu^k, \bar{\lambda}^k) &= g(x^k) - \mathcal{A}(x^k)^\top \hat{\lambda}^k \\ &= g(x^*) + \mathcal{H}(x^*)(x^k - x^*) - \mathcal{A}(x^*)^\top \hat{\lambda}^k \\ &\quad - \textstyle\sum_{j=1}^m \hat{\lambda}_j^k \mathcal{H}_j(x^*)(x^k - x^*) + r_1(x^k, (x^*, \hat{\lambda}^k) \\ &= g_L(x^*, \lambda^*) + H_L(x^*, \lambda^*)(x^k - x^*) - \mathcal{A}(x^*)^\top(\hat{\lambda}^k - \lambda^*) \\ &\quad + r_1(x^k, (x^*, \hat{\lambda}^k) + r_2(x^k, (x^*, \hat{\lambda}^k, \lambda^*), \end{aligned} \tag{5.70}$$

avec

$$r_1(x^k, x^*, \hat{\lambda}^k) = \int_0^1 (H_L(x^k + s(x^* - x^k), \hat{\lambda}^k) - H_L(x^*, \hat{\lambda}^k))(x^k - x^*) ds \tag{5.71}$$

et

$$r_2(x^k, x^*, \hat{\lambda}^k, \lambda^*) = \sum_{j=1}^m (\hat{\lambda}_j^k - \lambda^*) \mathcal{H}_j(x^*)(x^k - x^*). \tag{5.72}$$

Comme les matrices hessiennes de f et c_j sont bornées et Lipchitz dans le voisinage de x^* et que $\hat{\lambda}^k$ converge vers λ^* alors pour des constantes positives a_7 et a_8 :

$$\begin{aligned} r_1(x^k, x^*, \hat{\lambda}^k) &\leq a_7 \left\| x^k - x^* \right\|^2, \\ r_2(x^k, x^*, \hat{\lambda}^k, \lambda^*) &\leq a_8 \left\| x^k - x^* \right\| \left\| \hat{\lambda}^k - \lambda^* \right\|. \end{aligned} \tag{5.73}$$

De plus, par l'intermédiaire du développement de Taylor et du fait que $c(x^*) = 0$:

$$c\left(x^k\right) = \mathcal{A}(x^*)(x^k - x^*) + r_3(x^k, x^*) \tag{5.74}$$

avec

$$(r_3(x^k, x^*))_i = \int_0^1 s \int_0^1 (x^k - x^*)^\top H_i(x^* + ts(x^k - x^*))(x^k - x^*)\, dt\, ds \tag{5.75}$$

(voir [86], page 11).

Comme les matrices hessiennes de c_i sont bornées dans le voisinage x^*, alors :

$$r_3(x^k, x^*) \leq a_9 \|x^k - x^*\|^2, \tag{5.76}$$

pour une constante positive a_9. En combinant (5.70), (5.73), (5.74) et (5.75), on a :

$$\begin{bmatrix} \mathcal{H}_L(x^*,\lambda^*) & -\mathcal{A}(x^*)^\top \\ -\mathcal{A}(x^*) & 0 \end{bmatrix} \begin{bmatrix} x^k - x^* \\ \hat{\lambda}^k - \lambda^* \end{bmatrix} = \begin{bmatrix} \nabla\, L_D(x^k,\lambda^k,\mu^k,\bar{\lambda}^k) - g_L(x^*,\lambda^*) \\ -c(x^k) \end{bmatrix} - \begin{bmatrix} r_1 + r_2 \\ -r_3 \end{bmatrix}, \tag{5.77}$$

où les variables de $r_l,\ r_2$ et r_3 sont omises. Pour la suite, nous notons $y_{[J]}$ le vecteur formé en sélectionnant uniquement les composantes de y indexées par l'ensemble J. Nous pouvons réécrire (5.77) :

$$\begin{bmatrix} \mathcal{H}_L(x^*,\lambda^*)_{[I_F,I_F]} & \mathcal{H}_L(x^*,\lambda^*)_{[I_F,I_D]} & -\mathcal{A}(x^*)^\top_{[I_F]} \\ \mathcal{H}_L(x^*,\lambda^*)_{[I_D,I_F]} & \mathcal{H}_L(x^*,\lambda^*)_{[I_D,I_D]} & -\mathcal{A}(x^*)^\top_{[I_D]} \\ -\mathcal{A}(x^*)_{[I_F]} & -\mathcal{A}(x^*)_{[I_D]} & 0 \end{bmatrix} \begin{bmatrix} (x^k - x^*)_{[I_F]} \\ (x^k - x^*)_{[I_D]} \\ \hat{\lambda}^k - \lambda^* \end{bmatrix} = \begin{bmatrix} \nabla\, L_D(x^k,\lambda^k,\mu^k,\bar{\lambda}^k)_{[I_F]} \\ (\nabla\, L_D(x^k,\lambda^k,\mu^k,\bar{\lambda}^k) - g_L(x^*,\lambda^*)_{[I_D]} \\ -c(x^k) \end{bmatrix} - \begin{bmatrix} (r_1+r_2)_{[I_F]} \\ (r_1+r_2)_{[I_D]} \\ -r_3 \end{bmatrix}. \tag{5.78}$$

En enlevant les blocs du centre de (5.78) horizontalement, on obtient

$$\begin{bmatrix} \mathcal{H}_L(x^*,\lambda^*)_{[I_F,I_F]} & -\mathcal{A}(x^*)^\top_{[I_F]} \\ -\mathcal{A}(x^*)_{[I_F]} & 0 \end{bmatrix} \begin{bmatrix} (x^k - x^*)_{[I_F]} \\ \hat{\lambda}^k - \lambda^* \end{bmatrix} = \begin{bmatrix} \nabla\, L_D(x^k,\lambda^k,\mu^k,\bar{\lambda}^k)_{[I_F]} - \mathcal{H}_L(x^*,\lambda^*)_{[I_F,I_D]}(x^k)_{[I_D]} \\ -c(x^k) + \mathcal{A}(x^*)_{[I_D]}\ (x^k)_{[I_D]} \end{bmatrix} - \begin{bmatrix} (r_1+r_2)_{[I_F]} \\ -r_3 \end{bmatrix} \tag{5.79}$$

D'après les équations (5.39), (5.62) et

$$\left\| \nabla\, L_D(x^k,\lambda^k,\mu^k,\bar{\lambda}^k)_{[I_F]} \right\| \leq\ \omega^k, \tag{5.80}$$

adaptée de (5.40), nous avons :

$$\left\| x^k_{[I_D]} \right\| \leq\ \omega^k. \tag{5.81}$$

Donc, en utilisant (5.69) :

$$\left\|x^k - x^*\right\| \leq \left\|(x^k - x^*)_{[I_F]}\right\| + \ \omega^k. \tag{5.82}$$

Soit $\Delta\ x^k = \left\|(x^k - x^*)_{[I_F]}\right\|$. En utilisant (5.68) et (5.82) et en posant $a_{10} = a_1 + a_2$ nous avons :

$$\left\|\hat{\lambda}^k - \lambda^*\right\| \leq a_{10}\ \omega^k + a_2\ \Delta\ x^k. \tag{5.83}$$

De plus, les équations (5.73), (5.76), (5.82) et (5.83) donnent :

$$\left\|\begin{bmatrix}(r_1 + r_2)_{[I_F]} \\ r_3\end{bmatrix}\right\| \leq a_{11}(\Delta\ x^k)^2 + a_{12}\Delta\ x^k\ \omega^k + a_{13}(\ \omega^k)^2, \tag{5.84}$$

où $a_{11} = a_7 + a_9 + a_8\ a_2,\ a_{12} = 2(a_7 + a_9) + a_8(a_{10} + a_2)$, et $a_{13} = a_7 + a_9 + a_8\ a_{10}$. Aussi, d'après les équations (5.66), (5.81), (5.80), et (5.82),

$$\left\|\begin{bmatrix}\nabla\ L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)_{[I_F]} - \mathcal{H}_L(x^*, \lambda^*)_{[I_F, I_D]}(x^k)_{[I_D]} \\ c(x^k) - \mathcal{A}(x^*)_{[I_D]}\ (x^k)_{[I_D]}\end{bmatrix}\right\| \leq a_{14}\ \omega^k + \ \eta^k, \tag{5.85}$$

où $a_{14} = 1 + \left\|\begin{bmatrix}\mathcal{H}_L(x^*, \lambda^*)_{[I_F, I_D]} \\ \mathcal{A}(x^*)_{[I_D]}\end{bmatrix}\right\|$.

Par l'hypothèse 5, la matrice des coefficients sur le côté gauche du système (5.79) est inversible et son inverse a une norme $\mathcal{M}$. Nous pouvons obtenir donc :

$$\left\|\begin{bmatrix}(x^k - x^*)_{[I_F]} \\ \hat{\lambda}^k - \lambda^*\end{bmatrix}\right\| \leq \mathcal{M}[a_{14}\ \omega^k + \ \eta^k + a_{11}(\Delta\ x^k)^2 + a_{12}\Delta\ x^k\ \omega^k + a_{13}(\ \omega^k)^2]. \tag{5.86}$$

Nous choisissons k suffisamment grand pour que

$$\omega^k \leq \min\{1, 1/(2\mathcal{M}a_{12})\}. \tag{5.87}$$

Alors, les équations (5.86) et (5.87) donnent :

$$\Delta\ x^k \leq \frac{1}{2}\ \Delta\ x^k + \mathcal{M}\left(a_{15}\ \omega^k + \ \eta^k + a_{11}(\ \Delta^k)^2\right) \tag{5.88}$$

avec $a_{15} = a_{13} + a_{14}$. Comme $\Delta\ x^k$ converge vers zéro, nous avons pour tout k suffisamment grand :

$$\left\|\Delta\, x^k\right\| \leq 1/(4\mathcal{M}a_{11}), \tag{5.89}$$

Les inégalités (5.88) et (5.89) impliquent :

$$\Delta\, x^k \leq 4\,\mathcal{M}[a_{15}\,\omega^k + \,\eta^k]. \tag{5.90}$$

En posant $a_3 = 4\,\mathcal{M}a_{15} + 1$ et $a_4 = 4\,\mathcal{M}$ et en utilisant les équations (5.82) et (5.90), nous obtenons l'inégalité (5.63).

Selon l'hypothèse 1, il existe une constante $L^* > 0$ telle que :

$$\left\|c(x^k) - c(x^*)\right\| \leq L^* \left\|x^k - x^*\right\|. \tag{5.91}$$

En posant $a_6 = L^*\, a_3$ et $a_{16} = L^*\, a_4$, les équations (5.63) et (5.91) établissent (5.65).

Les relations (5.63) et (5.68) impliquent que (5.64) est satisfaite, avec $a_5 = a_1 + a_2\, a_3$ et $a_{17} = a_2\, a_4$. ■

Il est possible de démontrer facilement le corollaire suivant qui généralise le résultat (5.64) pour n'importe quelle estimation des multiplications de Lagrange qui satisfait une condition donnée :

Corollaire 1. *Considérons que les hypothèses du lemme 4 sont satisfaites et que* $\hat{lambda}^{k+1}$ *est une estimation des multiplicateurs de Lagrange qui satisfait :*

$$\left\|\hat{\lambda}^{k+1} - \lambda^*\right\| \leq a_{18}\left\|x^k - x^*\right\| + a_{19}\,\omega^k, \tag{5.92}$$

pour des constantes positives a_{18} *et* a_{19} *et pour* k *suffisamment grand. Il existe alors des constantes positives* a_{20} *et* a_{21} *et une valeur entière* k_0 *telle que l'équation* (5.63) *est satisfaite et pour tout* $k \geq k_0$ *:*

$$\left\|\hat{\lambda}^{k+1} - \lambda^*\right\| \leq a_{20}\,\omega^k + a_{21}\,\eta^k. \tag{5.93}$$

Démonstration. En utilisant les équations (5.63) et (5.92) et en posant $a_{20} = a_{18}\, a_3 + a_{19}$ et $a_{21} = a_{18}\, a_4$, nous retrouvons l'équation (5.93). ■

Nous introduisons une hypothèse supplémentaire relative aux écarts complémentaires stricts pour

profiter pleinement des variables flottantes.

Hypothèse 6. *Si les itérés $x^k, k \in K$, convergent vers un point limite x^* avec λ^* le vecteur des multiplicateurs de Lagrange, nous supposons que :*

$$\mathcal{J}_2 = \{i \in \{1, 2, \ldots, n\} : x_i^* = 0 \textit{ et } (\nabla L(x^*, \lambda^*))_i = 0\} \tag{5.94}$$

est vide.

Nous montrons maintenant que toute variable flottante est loin de ses bornes en x^*.

Théorème 4. *Supposons que la suite de points $\{x^k\}$ générée par l'algorithme 13 converge vers un point limite x^* et que les hypothèses 1, 2, 3 et 6 sont satisfaites. Alors, pour k suffisamment grand, l'ensemble des variables flottantes sont exactement celles dont les valeurs sont loin de leurs bornes en x^*.*

Démonstration. D'après le théorème 3, $\nabla L_D(x^k, \lambda^k, \mu^k, \bar{\lambda}^k)$ converge vers $\nabla L(x^*, \lambda^*)$ et selon le lemme 2, les variables de l'ensemble I_5 convergent vers zéro et les composantes correspondantes de $\nabla L(x^*, \lambda^*)$ sont égales à zéro. L'hypothèse 6 suggère donc que I_5 est vide. Ceci veut dire que, pour k suffisamment grand, chaque variable appartient aux ensembles I_1 ou I_2. Par définition, les variables dans I_1 sont flottantes et convergent vers une valeur loin de zéro alors que les variables dans I_2 sont dominées et convergent vers zéro. ■

Grâce au théorème 4, il est possible d'évaluer $\bar{\lambda}^k$, pour tout k suffisamment grand, par l'intermédiaire du système suivant :

$$\bar{\lambda}^k = -\left(\bar{A}^k\right)^+ \bar{\nabla}\ f^k \tag{5.95}$$

avec $\bar{A}^k$ et $\bar{\nabla}\ f^k$ les colonnes de $A(x^k)$ et les composantes de $\nabla\ f(x^k)$ respectivement correspondant aux variables flottantes en x^k.

Finalement, nous discutons de la vitesse de convergence de l'algorithme présenté dans ce chapitre. On dit qu'une suite $\{u_n\}$ converge R-linéairement vers u^* si :

$$\|u_n - u^*\| \leq \Lambda_n \tag{5.96}$$

avec $\{\Lambda_n\}$ une suite qui converge linéairement vers 0. L'algorithme 13 possède un taux de convergence R-linéaire.

Théorème 5. *Supposons que la suite de points $\{x^k\}$ générée par l'algorithme 13 converge vers un point limite x^* et que les hypothèses 1, 2, 3 et 6 sont satisfaites. Les itérés x^k, les estimations des multiplicateurs de Lagrange λ^k et n'importe quelle estimation $\hat{\lambda}^k$ satisfaisant le corollaire 1 convergent R-linéairement avec un taux d'au plus $\hat{\mu}^{\min(\beta_\omega,\beta_\eta)}$, où $\hat{\mu} = \min(\gamma_1, \mu)$ et où μ est la plus petite valeur du paramètre de pénalité générée par l'algorithme.*

Démonstration. Cette preuve ressemble à celle du lemme 4. Comme μ est borné loin de zéro, il existe une valeur limite μ telle que $\mu^k > \mu$. On pose $\hat{\mu} = \min\{\mu, \gamma_1\}$. À partir d'une itération k suffisamment grande, l'étape 4 de l'algorithme est toujours exécutée et les deux équations suivantes sont établies :

$$\omega^{(k+1)} = (\hat{\mu})^{\beta_\omega}\omega^k \text{ et } \eta^{k+1} = (\hat{\mu})^{\beta_\eta}\eta^k. \tag{5.97}$$

De plus, le lemme 4 suggère que :

$$\left\|x^k - x^*\right\| \leq a_3\,\omega^k + a_4\,\eta^k. \tag{5.98}$$

Comme $\beta_\eta < 1$, les équations (5.97) et (5.98) assurent que x^k converge vers x^* avec une vitesse R-linéaire d'un taux $\hat{\mu}^{\min(\beta_\omega,\beta_\eta)}$. Il est possible de prouver la même chose pour les suites $\{\hat{\lambda}^k\}^k$ et $\{\hat{\lambda}^k\}$ en utilisant les équations (5.68), (5.92) et (5.98). ■

5.4 Détails d'implémentation

Nous avons implémenté une version MATLAB de l'algorithme du Lagrangien augmenté en norme ℓ_1 qui traite des problèmes quadratiques avec contraintes quadratiques :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f(x) \\ \text{sujet à} & c_j(x) \leq 0, \quad j \in \{1, 2, \ldots, m\} \end{array} \tag{5.99}$$

Quand on traite uniquement des contraintes d'inégalités, à chaque itération de l'algorithme, il peut exister des contraintes qui ne sont ni ϵ-actives ni ϵ-violées vu que l'ensemble des contraintes ϵ-violées est redéfini par :

$$V_{\epsilon^k}(x^k) = \{j \in \{1, 2, \ldots, m\} : c_j(x^k) > \epsilon^k\}. \tag{5.100}$$

L'expression du Lagrangien augmenté en norme ℓ_1 devient alors :

$$\begin{aligned}\mathcal{L}_A^{\ell_1}(x^k,\lambda^k,\mu^k) &= f(x^k) - \sum_{j=1}^{m} \lambda_j^k c_j(x^k) + \tfrac{1}{\mu^k} \sum_{j=1}^{m} s_{jj}^k \max\{0, c_j(x^k)\} \\ &= f(x^k) - \sum_{j=1}^{m} \lambda_j^k c_j(x^k) + \tfrac{1}{\mu^k} \sum_{j\in V_{\epsilon^k}} s_{jj}^k c_j(x^k) + \tfrac{1}{\mu^k} \sum_{j\in A_{\epsilon^k}} s_{jj}^k \max\{0, c_j(x^k)\},\end{aligned} \tag{5.101}$$

et la mise à jour des multiplicateurs de Lagrange doit prendre en compte les contraintes dont les indices n'appartiennent pas à $A_{\epsilon^k} \cup V_{\epsilon^k}$:

$$\lambda_j^{k+1} = \begin{cases} \lambda_j^k + \bar{\lambda}_j^k & , \quad \text{si } j \in A_{\epsilon^k}, \\ \lambda_j^k - \frac{s_{jj}^k}{\mu^k} & , \quad \text{si } j \in V_{\epsilon^k}, \\ \lambda_j^k & , \quad \text{si } j \notin A_{\epsilon^k} \cup V_{\epsilon^k}, \end{cases} \tag{5.102}$$

où $\bar{\lambda}^k$ est un vecteur de multiplicateurs pour les contraintes ϵ-actives satisfaisant :

$$\nabla f(x^k) - \sum_{j=1}^{m} \lambda_j^k \nabla c_j(x^k) + \frac{1}{\mu^k} \sum_{j\in V_{\epsilon^k}} s_{jj}^k \nabla c_j(x^k) = \sum_{j\in A_{\epsilon^k}} \bar{\lambda}_j^k \nabla c_j(x^k). \tag{5.103}$$

En plus, l'implémentation de cette méthode prend en charge des contraintes de bornes généralisées (chaque variable x_i possède une borne inférieure l_i et une borne supérieure u_i). Il suffit de redéfinir la projection (5.35) de la manière suivante :

$$(\mathcal{P}_{\mathcal{X}}(x))_i = \begin{cases} l_i & \text{si } x_i \leq l_i, \\ u_i & \text{si } x_i \geq u_i, \\ x_i & \text{sinon.} \end{cases} \tag{5.104}$$

L'algorithme du Lagrangien augmenté en norme ℓ_1 peut être utilisé directement en mettant à jour la définition de la projection par celle donnée ci-dessus. Même les concepts de variables flottantes et dominées peuvent être généralisés pour retrouver les résultats de convergence prouvés dans ce chapitre.

Dans la méthode de Coleman et Conn, si la recherche linéaire principale le long de la direction h^k échoue à introduire une dimunition suffisante dans la valeur de $\mathcal{L}_A^{\ell_1}$, une recherche linéaire secondaire basée sur une interpolation cubique est effectuée dans la direction h^k. Dans le chapitre précédent, nous avons substitué cette sous-routine secondaire dans l'algorithme 7 par une recherche linéaire avec une interpolation quadratique et nous décidons de garder ce choix pour traiter des problèmes quadratiques avec contraintes quadratiques.

La recherche linéaire le long de h^k cherche à mettre à jour l'itéré x^k tout en maintenant les valeurs

de λ^k et μ^k fixées. Lorsque la recherche linéaire primaire décrite dans l'algorithme 7 échoue, la recherche linéaire par interpolation quadratique essaye de trouver un pas positif α dans un intervalle $[a, c]$ $(0 \leq a < c)$ qui minimise la fonction suivante :

$$\begin{aligned} \varphi(\alpha) &= D(x^k + \alpha h^k, \lambda^k, \mu^k) \\ &= f(x^k + \alpha h^k) - \sum_{j=1}^{m} \lambda_j^k c_j(x^k + \alpha h^k) + \frac{1}{\mu^k} \sum_{j \in V_{\epsilon^k}} s_{jj}^k c_j(x^k + \alpha h^k). \end{aligned} \tag{5.105}$$

Il faut remarquer qu'on considère uniquement la partie différentiable de $\mathcal{L}_A^{\ell_1}$ vu que la direction h^k est supposée, par définition, ne pas affecter les contraintes ϵ-actives. La recherche linéaire par interpolation quadratique est une approche itérative qui se base sur les valeurs a et c de l'intervalle $[a, c]$ et une valeur b satisfaisant $a < b < c$ pour construire, à chaque itération, un polynôme quadratique approchant la fonction φ. Si un pas α minimise le polynôme quadratique, alors la comparaison entre les valeurs de φ en α et en a, b et c permet de réduire et mettre à jour l'intervalle de recherche. L'algorithme 15 décrit la méthode de recherche linéaire par interpolation quadratique qui retourne un pas α pour mettre à jour l'itéré x^k.

Finalement, un détail d'implémentation important concerne la mise à jour de l'approximation de la projection de la matrice hessienne $(Z^k)^\top B^k Z^k$ tout en la gardant définie positive. Par exemple, pour les méthodes quasi-Newton, il existe plusieurs approches telles que les formules DFP [65] (Davidon, Fletcher et Powell) et BFGS (Broyden [36], Fletcher [63], Goldfarb [76] et Shanno [153]) qui permettent de faire cette mise à jour (voir chapitre 8 de [136]). Dans notre cas, on s'intéresse principalement à des problèmes quadratiques avec contraintes quadratiques et l'évaluation des matrices hessiennes dans ce cas est facile.

Nous comparons entre deux versions de l'algorithme. La première utilise la projection de la matrice hessienne $(Z^k)^\top H^k Z^k$ du Lagrangien augmenté en norme ℓ_1 directement pour calculer la direction de descente h^k. Les conditions suffisantes de second ordre assurent que la matrice $(Z^k)^\top H^k Z^k$ devient définie positive dans le voisinage d'un minimum local de D. Quand l'itéré courant est loin d'un minimum local, il est possible que la matrice $(Z^k)^\top H^k Z^k$ ne soit pas définie positive et que la direction h^k ne soit pas une direction de descente ($\nabla D(x^k, \lambda^k, \mu^k)^\top h^k > 0$). Dans ce cas, il suffit de considérer la direction $-h^k$ qui est forcément une direction de descente pour D.

Pour la seconde version, si la projection de la matrice hessienne $(Z^k)^\top H^k Z^k$ du Lagrangien augmenté est définie positive, nous l'utilisons directement sans évaluer une approximation. Cependant, si elle n'est pas définie positive, nous allons utiliser une analyse spectrale de $(Z^k)^\top H^k Z^k$ pour déterminer les valeurs propres (regroupées dans une matrice diagonale Σ) et les vecteurs propres (qui

Algorithme 15: Recherche linéaire par interpolation quadratique

$0 < \delta << 1$
$\varphi_a \leftarrow \varphi(a) = D(x^k + ah^k, \lambda^k, \mu^k)$
$\varphi_b \leftarrow \varphi(b) = D(x^k + bh^k, \lambda^k, \mu^k)$
$\varphi_c \leftarrow \varphi(c) = D(x^k + ch^k, \lambda^k, \mu^k)$
$\alpha \leftarrow \frac{b+c}{2}$
$\varphi_\alpha \leftarrow \varphi(\alpha) = D(x^k + \alpha h^k, \lambda^k, \mu^k)$
Tant que $|\alpha - b| < \delta$
  Si $(\alpha - b)(\alpha - c) < 0$
    Si $\varphi_\alpha < \varphi_2$
      $a \leftarrow b$; $\varphi_a \leftarrow \varphi_b$
      $b \leftarrow \alpha$; $\varphi_b \leftarrow \varphi_\alpha$
    Sinon
      $c \leftarrow \alpha$; $\varphi_c \leftarrow \varphi_\alpha$
    fin Si
  Sinon
    Si $\varphi_\alpha < \varphi_2$
      $c \leftarrow b$; $\varphi_c \leftarrow \varphi_b$
      $b \leftarrow \alpha$; $\varphi_b \leftarrow \varphi_\alpha$
    Sinon
      $a \leftarrow \alpha$; $\varphi_a \leftarrow \varphi_\alpha$
    fin Si
  fin Si
  $H \leftarrow 2(\varphi_a(b-c) + \varphi_b(c-a) + \varphi_c(a-b))$
  Si $|H| < \delta$
    $\alpha \leftarrow b$; $\varphi_\alpha \leftarrow \varphi_b$
  Sinon
    $\alpha \leftarrow \frac{\varphi_a(b^2-c^2)+\varphi_b(c^2-a^2)+\varphi_c(a^2-b^2)}{H}$
    Si $(\alpha - a)(\alpha - c) < 0$
      $\varphi_\alpha \leftarrow \varphi(\alpha) = D(x^k + \alpha h^k, \lambda^k, \mu^k)$
    Sinon
      $\alpha \leftarrow b$; $\varphi_\alpha \leftarrow \varphi_b$
    fin Si
  fin Si
fin Tant que

constituent les colonnes d'une matrice P) de cette matrice :

$$(Z^k)^\top H^k Z^k = P \Sigma P^\top. \tag{5.106}$$

On définit la matrice diagonale Σ' grâce aux éléments σ'_{ii} qui satisfont :

$$\sigma'_{ii} = \max\{\sigma_{ii}, \tau\}, \tag{5.107}$$

ou σ_{ii} représente les éléments de la diagonale de Σ et $0 < \tau \ll 1$. On pose :

$$(Z^k)^\top B^k Z^k = P\Sigma' P^\top, \tag{5.108}$$

qui est clairement définie positive. Cette approche ne donne pas la meilleure approximation possible, mais elle est simple à implémenter et n'est pas très coûteuse pour des problèmes de petite dimension. La direction h^k est obtenue par la formule $h^k = Z^k w$ où w est la solution de :

$$((Z^k)^\top B^k Z^k)w = -(Z^k)^\top \nabla D. \tag{5.109}$$

Quand $(Z^k)^\top B^k Z^k$ est définie positive, il est possible d'utiliser la décomposition LDL pour inverser cette matrice et évaluer la direction de recherche h^k.

Le tableau 5.1 compare les deux versions du Lagrangien augmenté en norme ℓ_1 décrit dans cette section : la version 1 utilise les vraies matrices hessiennes du problème alors que la version 2 utilise une approximation définie positive des matrices hessiennes. La comparaison est conduite sur un ensemble de 12 problèmes quadratiques avec contraintes quadratiques de petite dimension choisies de [94].

Sans faire d'approximation des matrices hessiennes, la version 1 de l'algorithme arrive à trouver la solution globale pour tous les problèmes. La version 2 retrouve la solution globale pour 10 problèmes seulement et échoue à fournir une solution réalisable pour HS23. Cet échec est dû au fait que le nombre de contraintes dépasse le nombre de variables, ce qui augmente le risque de se retrouver avec un itéré dégénérescent. Nous n'avons pas encore traité cet aspect dans l'algorithme, mais il semble que la version 1 arrive à éviter ce problème avec l'instance HS23. Il arrive, cependant, que la version 1 se retrouve dans la même situation comme, par exemple, avec le problème HS44 qui se compose de 4 variables et de 6 contraintes. Le problème HS44 est absent du tableau 5.1 parce que les deux versions échouent à trouver une solution réalisable.

Tableau 5.1 Comparaison du Lagrangien augmenté en norme ℓ_1 avec (version 2) et sans (version 1) l'utilisation de l'approximation de la matrice hessienne B^k.

Problèmes			*Point de départ*			*Solution globale*		$\ell_1 AUGLAG$ (*version* 1)		$\ell_1 AUGLAG$ (*version* 2)	
Nom	n	m	x^0	$f(x^0)$	*Réalisable?*	x^*	$f(x^*)$	x^*	$f(x^*)$	x^*	$f(x^*)$
$HS3$	2	0	$(10, 1)$	1.00081	*oui*	$(0, 0)$	0	$(0, 0)$	0	$(0, 0)$	0
$HS10$	2	1	$(-10, 10)$	-20	*non*	$(0, 1)$	-1	$(0, 1)$	-1	$(0, 1)$	-1
$HS11$	2	1	$(4.9, 0.1)$	-24.98	*non*	$(1.2348, 1.5247)$	-8.4985	$(1.2348, 1.5247)$	-8.4985	$(1.2348, 1.5247)$	-8.4985
$HS12$	2	1	$(0, 0)$	0	*oui*	$(2, 3)$	-30	$(2, 3)$	-30	$(2, 3)$	-30
$HS18$	2	2	$(2, 2)$	4.04	*non*	$(15.811, 1.5811)$	5	$(15.811, 1.5811)$	5	$(15.811, 1.5811)$	5
$HS21$	2	1	$(-1, -1)$	-98.99	*non*	$(2, 0)$	-99.96	$(2, 0)$	-99.96	$(2, 0)$	-99.96
$HS22$	2	2	$(2, 2)$	1	*non*	$(1, 1)$	1	$(1, 1)$	1	$(1, 1)$	1
$HS23$	2	5	$(3, 1)$	10	*non*	$(1, 1)$	2	$(1, 1)$	2	-	-
$HS30$	3	1	$(1, 1, 1)$	3	*oui*	$(1, 0, 0)$	1	$(1, 0, 0)$	1	$(1, 0.01562, 0)$	1.00024
$HS31$	3	1	$(1, 1, 1)$	19	*oui*	$(0, 57735, 1.7321, 0)$	6	$(0, 57735, 1.7321, 0)$	6	$(0, 57735, 1.7321, 0)$	6
$HS35$	3	1	$(0.5, 0.5, 0.5)$	2.25	*oui*	$(1.3333, 0.77778, 0.44444)$	0.11111	$(1.3333, 0.77778, 0.44444)$	0.11111	$(1.3333, 0.77778, 0.44444)$	0.11111
$HS65$	3	1	$(-5, 5, 0)$	136.11	*non*	$(3.6505, 3.6505, 4.6204)$	0.95353	$(3.6505, 3.6505, 4.6204)$	0.95353	$(3.6505, 3.6505, 4.6204)$	0.95353

5.5 Discussion

Dans ce chapitre, nous avons introduit une nouvelle méthode du Lagrangien augmenté qui utilise un terme de pénalité en norme ℓ_1 au lieu de la norme quadratique usuelle. Nous avons rappelé d'importants résultats de convergence pour l'approche de Coleman et Conn qui régit la boucle interne de l'algorithme responsable de mettre à jour l'itéré courant de notre nouvelle méthode. De plus, nous avons développé une analyse de convergence pour le Lagrangien augmenté en norme ℓ_1 qui établit la convergence globale et le taux de convergence R-linéaire de la méthode.

Nous avons présenté également des détails d'implémentation de cette méthode lors du traitement des contraintes d'inégalités directement et quelques résultats préliminaires pour valider notre implémentation. Nos projets futurs impliquent l'implémentation d'un système de mise à jour de l'approximation de la matrice hessienne qui préserve sa structure définie positive ainsi que la construction d'un banc de test pour mettre les implémentations réalisées à l'épreuve.

Dans des travaux futurs, sortant du cadre de cette thèse, nous visons à implémenter deux autres versions du Lagrangien augmenté en norme ℓ_1 pour la résolution des problèmes quadratiques avec contraintes quadratiques. La première version utilise des variables d'écart pour transformer les contraintes d'inégalité en contraintes d'égalité et la seconde version remplace la méthode de Coleman et Conn dans la boucle interne par une approche de région de confiance. Le but est d'explorer la possibilité d'implémenter une version ℓ_1AUGLAG orientée pour les QCQP de grande taille.

CHAPITRE 6 UNE NOUVELLE STRATÉGIE DE SÉLECTION DE VARIABLES POUR PSD-MADS

Les chapitres 4 et 5 traitaient respectivement de la résolution d'un sous-problème par une méthode quadratique, et son analyse de convergence. Revenons maintenant à l'optimisation sans dérivées. Les travaux de ce chapitre correspondent à un article soumis dans *Mathematical Programming Computation*. Nous considérons le problème suivant :

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & f(x) = c_0(x) \\ \text{sujet à} & c_j(x) \leq 0, \qquad j \in \{1, 2, \ldots, m\} \end{array} \tag{6.1}$$

où $\mathcal{X}$ est un sous-ensemble de $\mathbb{R}^n$, $(c_j)_{j\in\{0,1,\ldots,m\}}$ sont des fonctions à valeurs réelles alors que m et n sont des entiers positifs. Comme pour la section précédente, nous supposons que la fonction objectif et les contraintes sont des boites noires. Nous utilisons la notation c_0 pour désigner la fonction objectif au lieu de f pour uniformiser les notations et insister sur le fait que l'étude de l'influence des variables est conduite sur une sortie d'un problème, que ce soit l'objectif ou une contrainte. De plus, on considère que la dimension n du problème est assez grande étant donné le contexte DFO. Cette section considère des problèmes entre 500 et $4{,}000$ variables.

Les méthodes DFO considèrent typiquement des problèmes de petite taille. Par exemple, NOMAD [109] est recommandé pour des problèmes de moins de 20 variables. Pour des problèmes de plus grande taille, d'autres techniques sont nécessaires telles que le parallélisme.

Avec l'évolution rapide des technologies, tout le monde a accès à du matériel parallèle. Les logiciels et les algorithmes ont tendance à utiliser le parallélisme pour économiser le temps d'exécution et traiter des problèmes plus grands et plus complexes.

Ce travail a pour but d'améliorer la décomposition parallèle de l'espace de l'algorithme MADS (PSD-MADS [20]). PSD-MADS, décrit dans la section 2.8, décompose le problème d'optimisation (6.1) en une série de sous-problèmes de plus petite taille qui sont pris en charge par l'algorithme MADS d'une manière parallèle. La création des sous-problèmes se fait en sélectionnant aléatoirement un petit nombre de variables qui peuvent changer de valeurs alors que le reste des variables sont fixées à leurs valeurs courantes. L'objectif principal de ce chapitre est de trouver une manière plus efficace pour sélectionner les variables. Alors que la version originelle de PSD-MADS considérait des problèmes jusqu'à 500 variables, nous utilisons la nouvelle approche pour effectuer des tests sur des problèmes de taille allant jusqu'à $4{,}000$ variables.

Les remarques concluant l'article [20] suggèrent que la méthode de sélection de variables peut être

améliorée. Nous proposons une version améliorée de PSD-MADS qui se base sur des outils statistiques pour déterminer l'influence de chaque variable sur les sorties du problème et construit une matrice nommée la matrice de sensibilité. Nous introduisons différentes stratégies pour exploiter la matrice de sensibilité efficacement et nous utilisons une méthode de classification pour regrouper les variables avec une influence similaire dans la même grappe. De plus, l'algorithme de *clustering* va permettre de choisir la taille des sous-problèmes automatiquement.

Les sections 6.1 et 6.2 présentent les nouvelles méthodes de sélection de variables utilisées dans les versions améliorées de PSD-MADS. Ces algorithmes dépendent de cinq paramètres notés $\mathcal{P}_{vr}, \mathcal{P}_K, \mathcal{P}_S, \mathcal{P}_{ns}$ et $\mathcal{P}_{bbe}$. La section 6.3 décrit et reporte les résultats numériques qui permettent de déterminer des valeurs par défaut pour les nouveaux paramètres et de valider les méthodes améliorées. Une discussion s'ensuit dans la section 6.4.

6.1 Une méthode améliorée de sélection de variables

Le but principal de ce chapitre est de concevoir une nouvelle stratégie de sélection des variables importantes pour guider la décomposition parallèle en sous-problèmes.

6.1.1 Construction de la matrice de sensibilité

Pour déterminer les variables les plus influentes d'un problème, la nouvelle approche se base sur les travaux de [5] qui estiment les coefficients de sensibilité des variables par rapport à l'objectif. Ces estimateurs peuvent se généraliser pour quantifier l'influence des variables sur les contraintes.

Pour chaque variable x_i ($i \in \{1, 2, \ldots, n\}$) et pour chaque sortie c_j ($j \in \{0, 1, \ldots, m\}$), le coefficient de sensibilité est obtenu en calculant le quotient de la variation de c_j qui peut être expliquée par la corrélation linéaire entre c_j et x_i par rapport à la variation totale de c_j. Ce coefficient est estimé grâce à la formule suivante :

$$\tilde{s}_{ij} = \frac{\sum_{\ell=1}^{r_i} \left|A_i^\ell\right| ((\bar{c}_j)_i^\ell - \bar{c}_j)^2}{\sum_{x \in V} (c_j(x) - \bar{c}_j)^2} \tag{6.2}$$

où :

$$\bar{c}_j = \frac{1}{|V|} \sum_{x \in V} c_j(x) \qquad \text{et} \qquad (\bar{c}_j)_i^\ell = \frac{1}{\left|A_i^\ell\right|} \sum_{x \in A_i^\ell} c_j(x), \tag{6.3}$$

avec V l'ensemble des points de la cache, r_i le nombre de valeurs distinctes que prend x_i dans la

cache et A_i^ℓ l'ensemble des points pour lesquels la variable x_i prend sa $\ell^{\text{ème}}$ valeur parmi les r_i valeurs possibles. Par exemple, considérons la fonction à deux variables $f(x_1, x_2) = x_1 + 4x_2$ et une cache constitué de 6 points (tableau 6.1) :

La valeur moyenne de la fonction sur l'ensemble des 6 points est :

$$\bar{f} = \frac{1}{6}(0 + 1 - 5 + 1 + 5 + 4) = 1. \tag{6.4}$$

La variable x_1 prend 3 valeurs différentes dans la cache (-1, 0 et 1). Les ensembles de points et la moyenne de f pour chacune des valeurs sont donnés par :

$$\begin{array}{lcllcl} A_1^{-1} & = & \{(-1,-1),(-1,0.5)\}, & \bar{f}_1^{-1} & = & \frac{1}{2}(-5+1) = -2, \\ A_1^{0} & = & \{(0,0),(0,1)\}, & \bar{f}_1^{0} & = & \frac{1}{2}(0+4) = 2, \\ A_1^{1} & = & \{(1,0),(1,1)\}, & \bar{f}_1^{1} & = & \frac{1}{2}(1+5) = 3. \end{array} \tag{6.5}$$

En appliquant l'équation (6.2), le coefficient de sensibilité de la variable x_1 par rapport à la fonction f est :

$$\tilde{s}_1 = \frac{2(-2-1)^2 + 2(2-1)^2 + 2(3-1)^2}{(0-1)^2 + (1-1)^2 + (-5-1)^2 + (1-1)^2 + (5-1)^2 + (4-1)^2} = \frac{28}{62} = 0.452. \tag{6.6}$$

La variable x_2 prend 4 valeurs différentes dans la cache (-1, 0, 0.5 et 1). Les ensembles de points et la moyenne de f pour chacune des valeurs sont donnés par :

$$\begin{array}{lcllcl} A_2^{-1} & = & \{(-1,-1)\}, & \bar{f}_2^{-1} & = & \frac{1}{1}(-5) = -5, \\ A_2^{0} & = & \{(0,0),(1,0)\}, & \bar{f}_2^{0} & = & \frac{1}{2}(0+1) = 0.5, \\ A_2^{\frac{1}{2}} & = & \{(-1,0.5)\}, & \bar{f}_2^{\frac{1}{1}} & = & \frac{1}{1}(1) = 1, \\ A_2^{1} & = & \{(0,1),(1,1)\}, & \bar{f}_2^{1} & = & \frac{1}{2}(4+5) = 4.5. \end{array} \tag{6.7}$$

Tableau 6.1 Liste des points de la cache pour la fonction f

x_1	x_2	$f(x_1, x_2) = x_1 + 4x_2$
0	0	0
1	0	1
-1	-1	-5
-1	0.5	1
1	1	5
0	1	4

En appliquant l'équation (6.2), le coefficient de sensibilité de la variable x_2 par rapport à la fonction f est :

$$\tilde{s}_2 = \frac{(-5-1)^2 + 2(0.5-1)^2 + (1-1)^2 + 2(4.5-1)^2}{(0-1)^2 + (1-1)^2 + (-5-1)^2 + (1-1)^2 + (5-1)^2 + (4-1)^2} = \frac{61}{62} = 0.984. \quad (6.8)$$

Comme $\tilde{s}_2 > \tilde{s}_1$, alors la variable x_2 est plus influente que la variable x_1 sur la fonction f.

D'un point de vue pratique, il est difficile de mémoriser toutes les valeurs qu'une variable prend dans la cache pour des raisons de précision numérique de la comparaison décimale. Lors de la comparaison de deux valeurs décimales, une erreur $\epsilon > 0$ est introduite. Deux nombres a et b sont numériquement égaux si :

$$|b - a| < \epsilon. \quad (6.9)$$

Si ϵ est choisi trop petit, il est possible de se retrouver avec différentes valeurs de x_i pour tous les éléments de la cache, ce qui donne un coefficient de sensibilité égale à 1. Une option possible est de définir un paramètre $\mathcal{P}_{vr}$ où $10^{\mathcal{P}_{vr}}$ représente le nombre d'intervalles de valeurs des variables. Ce paramètre divise l'intervalle initial d'une variable en $10^{\mathcal{P}_{vr}}$ petits intervalles de même taille et au lieu de traquer toutes les valeurs spécifiques d'une variable dans la cache, l'idée est de pister les petits intervalles créés pour estimer le coefficient de sensibilité. La nouvelle formule pour estimer le coefficient de sensibilité est similaire à (6.2) excepté que $r_i = 10^{\mathcal{P}_{vr}}$ est le nombre d'intervalles et A_i^ℓ est l'ensemble de points pour lesquels la variable x_i appartient au $\ell^{\text{ème}}$ intervalle :

$$\tilde{s}_{ij} = \frac{\sum_{\ell=1}^{10^{\mathcal{P}_{vr}}} \left|A_i^\ell\right| ((\bar{c}_j)_i^\ell - \bar{c}_j)^2}{\sum_{x \in V} (c_j(x) - \bar{c}_j)^2}. \quad (6.10)$$

Quand $\mathcal{P}_{vr}$ tend à l'infini, les intervalles de valeurs sont réduits à des singletons ce qui revient à la formule (6.2). Ces coefficients de sensibilité ont une valeur entre 0 et 1 : plus $\tilde{s}_{ij}$ est proche de 1, plus la variable x_i est influente sur la réponse c_j. Nous créons une matrice de taille $n \times (m+1)$ regroupant tous les estimateurs $\tilde{s}_{ij}$ nommée matrice de sensibilité :

$$\tilde{S} = \begin{bmatrix} \tilde{s}_{10} & \tilde{s}_{11} & \cdots & \tilde{s}_{1m} \\ \tilde{s}_{20} & \tilde{s}_{21} & \cdots & \tilde{s}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{s}_{n0} & \tilde{s}_{n1} & \cdots & \tilde{s}_{nm} \end{bmatrix} \in \mathbb{R}^{n \times (m+1)}. \quad (6.11)$$

Plusieurs tests préliminaires ont été conduits pour vérifier que cette matrice de sensibilité reflète bien l'effet des variables sur les sorties du problème. À titre illustratif, considérons un programme linéaire afin de mettre en évidence les effets des variables :

$$
\begin{array}{lrcrcrcrcr}
\min\limits_{x\in\mathbb{R}^4} & & x_1 & + & x_2 & + & x_3 & + & x_4 & \\
 & 7x_1 & & & & + & 2x_3 & & & \leq \quad 1 \\
\text{sujet à :} & & & & x_2 & & & & & \leq \quad 5 \\
 & - & x_1 & - & x_2 & - & x_3 & - & x_4 & \leq -8 \\
 & - & x_1 & + & x_2 & - & 2x_3 & + & 5x_4 & \leq \quad 7
\end{array}
\tag{6.12}
$$

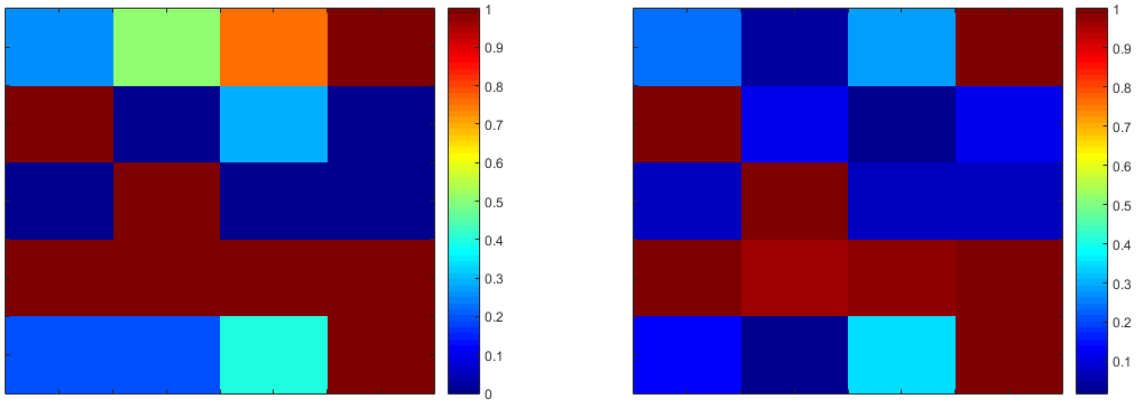
Nous évaluons les matrices de sensibilité pour cet exemple en générant un grand nombre de points n_{lhs} par échantillonnage en hypercubes Latins (nous testons 10^5, 10^6 et 10^7 comme valeurs de n_{lhs}). Les matrices de sensibilité devraient converger vers les coefficients du modèle en augmentant les valeurs $\mathcal{P}_{vr}$ et n_{lhs}. Pour confirmer ceci, nous utilisons des cartes d'activité pour représenter l'influence relative des variables obtenue par les matrices de sensibilité. Une carte d'activité (ou carte thermique) et une représentation graphique d'une matrice où chaque donnée de cette matrice est représentée par une nuance de couleur en accord avec sa valeur. Dans les figures 6.1 et 6.2, le spectre des couleurs utilisées s'étend entre le bleu et le rouge correspondant à l'intervalle des valeurs entre 0 et 1 que les coefficients de sensibilité peuvent prendre.

La figure 6.1 accroît la valeur n_{lhs} tout en fixant $\mathcal{P}_{vr} = 3$. Dans la figure 6.2, nous fixons $n_{lhs} = 10^6$, varions le paramètre $\mathcal{P}_{vr}$ entre 2 et 5 et comparons les matrices de sensibilité avec le modèle linéaire précédent.

Les figures 6.1 et 6.2 illustrent le fait que les matrices de sensibilité approximent assez bien le modèle linéaire. De plus, l'augmentation des deux valeurs $\mathcal{P}_{vr}$ et n_{lhs} semble rapprocher les matrices vers la sensibilité relative des variables du modèle linéaire.

Pour la nouvelle approche de sélection de variables, on garde le même cadre algorithmique que la méthode originelle sauf qu'on utilise cette matrice de sensibilité pour créer des groupes de variables qui sont rajoutés à une liste d'attente. À chaque fois qu'un processus esclave est libre, le serveur cache sélectionne un groupe de variables de la liste d'attente et la transmet au processus esclave en question. Quand la liste d'attente est vide, la matrice de sensibilité est mise à jour et de nouveaux groupes de variables sont rajoutés à la file.

Plusieurs stratégies peuvent être définies pour exploiter efficacement la matrice de sensibilité et en extraire les informations nécessaires pour construire les groupes de variables. Il est possible de considérer un ensemble de données $\{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$ où l'entrée $\tilde{s}_i = [\tilde{s}_{i0}\ \tilde{s}_{i1}\ \ldots\ \tilde{s}_{im}]^\top$ pour chaque variable x_i ($i \in \{1, 2, \ldots, n\}$) correspond aux coefficients de sensibilité de x_i pour les différentes



(a) modèle linéaire

(b) $\mathcal{P}_{vr} = 3; n_{lhs} = 10^5$

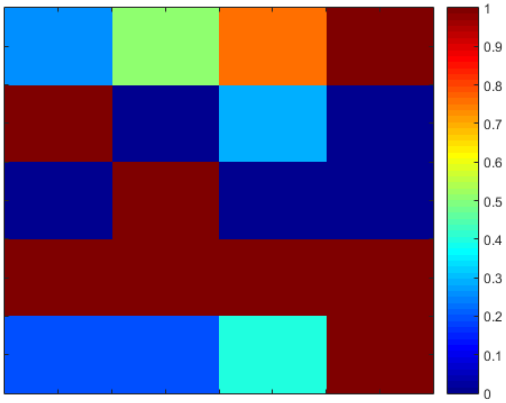
(c) $\mathcal{P}_{vr} = 3; n_{lhs} = 10^6$

(d) $\mathcal{P}_{vr} = 3; n_{lhs} = 10^7$

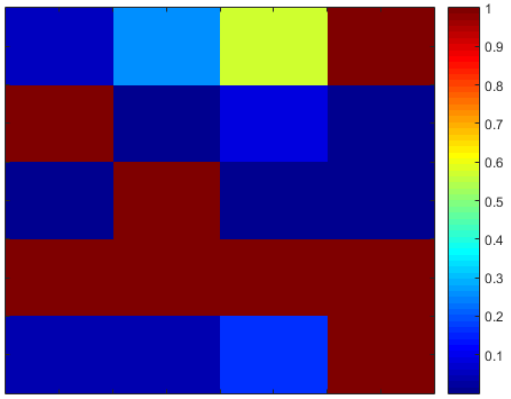
Figure 6.1 Effet de l'augmentation du nombre de points pour la construction de la matrice de sensibilité.

sorties du problème. Une méthode d'apprentissage non supervisée peut classer les données dans plusieurs groupes. Le tri de ces groupes par rapport à la distance de leurs centroïdes au point $[1\ 1\ \ldots\ 1]^\top$ permet de dévoiler les groupes plus proches de ce point comme les plus influents (vu que plus la sensibilité se rapproche de 1, plus la variable est influente sur une sortie donnée). Cette stratégie est très importante quand le nombre de ressources est limité. Cependant, nos tests sont effectués sur un super ordinateur de grande taille et comme le but est d'utiliser le plus de processeurs possible, tous les groupes de variables retournés par la méthode sont ajoutés à la file d'attente.

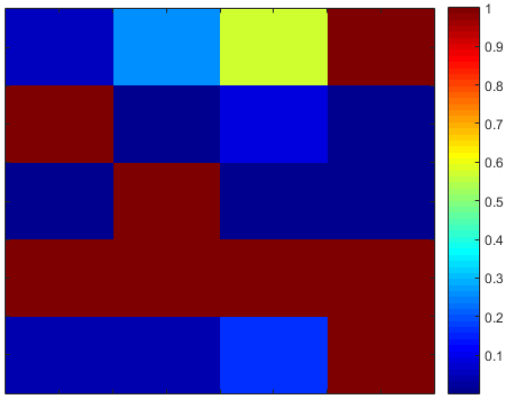
Il est aussi possible de considérer uniquement une sous-matrice de $\tilde{S}$ en sélectionnant un sous-ensemble de sorties. Par exemple, nous pouvons choisir le sous-ensemble de contraintes violées (à



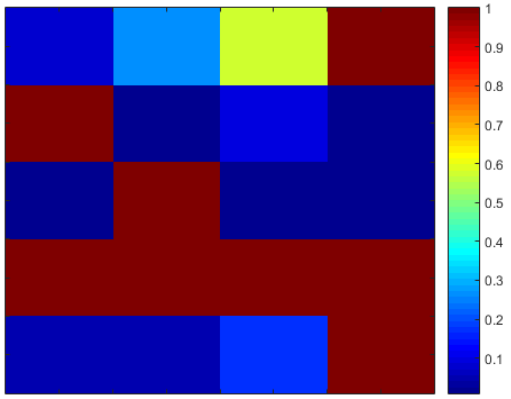
(a) modèle linéaire



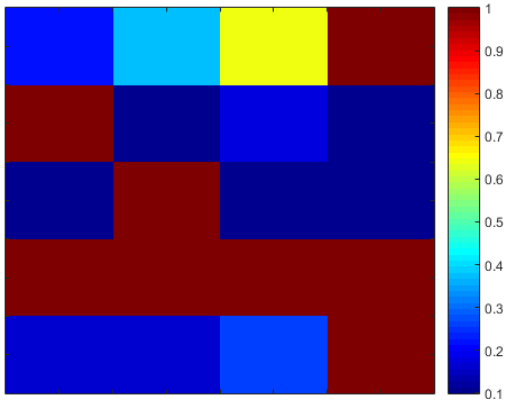
(b) $\mathcal{P}_{vr} = 2; n_{lhs} = 10^6$



(c) $\mathcal{P}_{vr} = 3; n_{lhs} = 10^6$



(d) $\mathcal{P}_{vr} = 4; n_{lhs} = 10^6$



(e) $\mathcal{P}_{vr} = 5; n_{lhs} = 10^6$

Figure 6.2 Effet du paramètre $\mathcal{P}_{vr}$ sur la matrice de sensibilité.

l'itéré courant), d'un côté, et nous concentrer sur les groupes de variables les plus influents, car ils peuvent potentiellement satisfaire ces contraintes. D'un autre côté, avec l'ensemble de contraintes satisfaites, les groupes de variables les moins influents ont plus de chance d'améliorer l'objectif sans altérer la réalisabilité des contraintes. Tous nos problèmes tests dans la section de résultats numériques utilisent un point de départ réalisable, ce qui veut dire que la décomposition des sorties en contraintes satisfaites et violées n'est pas adoptée.

Nous allons définir les quatre stratégies S1, S2, S3, S4 qui sont distinguées par le paramètre $\mathcal{P}_S$:

- La stratégie S1 utilise une technique de classification sur $\{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$ avec :

$$\tilde{s}_i = [\tilde{s}_{i0}\ \tilde{s}_{i1}\ \ldots\ \tilde{s}_{im}]^\top \text{ pour } i \in \{1, 2, \ldots, n\}. \tag{6.13}$$

- La stratégie S2 utilise deux fois l'algorithme de classification pour alimenter la file d'attente : la première fois sur l'ensemble relatif à la fonction objectif $\{\tilde{s}_{10}, \tilde{s}_{20}, \ldots, \tilde{s}_{n0}\}$ et, ensuite, sur l'ensemble relatif à toutes les contraintes $\{\tilde{s}_{1[1,m]}, \tilde{s}_{2[1,m]}, \ldots, \tilde{s}_{n[1,m]}\}$ avec :

$$\tilde{s}_{i[1,m]} = [\tilde{s}_{i1}\ \tilde{s}_{i2}\ \ldots\ \tilde{s}_{im}]^\top \text{ pour } i \in \{1, 2, \ldots, n\}. \tag{6.14}$$

- La stratégie S3 est identique à la stratégie S2 excepté que la seconde utilisation de l'algorithme de classification est appliquée à l'ensemble de données décrit dans la stratégie S1.
- La stratégie S4 applique la méthode de classification sur $\{\tilde{s}_{1\{j_1,j_2\}}, \tilde{s}_{2\{j_1,j_2\}}, \ldots, \tilde{s}_{n\{j_1,j_2\}}\}$ avec :

$$\tilde{s}_{i\{j_1,j_2\}} = [\tilde{s}_{ij_1}\ \tilde{s}_{ij_2}]^\top \text{ pour } i \in \{1, 2, \ldots, n\} \tag{6.15}$$

  où j_1 et j_2 sont deux indices de sorties choisis aléatoirement.

6.1.2 L'algorithme k-*mean* pour exploiter la matrice de sensibilité

L'apprentissage non supervisé est une branche de l'apprentissage automatique qui a pour objectif de trouver une structure dans un ensemble de données sans étiquette. L'algorithme k-*mean* [98] est probablement la méthode la plus populaire en classification non supervisée. Elle est utilisée pour classer les données en k groupes.

Chaque groupe contient des variables qui ont des caractéristiques similaires. Dans notre cas, une grappe contient les variables avec plus ou moins la même influence sur le problème. Ces groupes sont rajoutés à la liste d'attente et à chaque fois qu'un processus esclave a besoin d'un nouveau sous-problème, un groupe de variables de la liste d'attente lui est envoyé.

Algorithme 16: L'algorithme k-*mean* [98]

1: Initialiser k centroïdes de groupes en utilisant un échantillonnage via hypercube Latin.
2: Assigner chaque entrée de l'ensemble de données au groupe avec le centroïde le plus proche.
3: Mettre à jour les positions des centroïdes.
4: Répéter 2 et 3 jusqu'à ce que les centroïdes demeurent en position fixe.

L'algorithme k-*mean* est une approche locale vu que, selon l'initialisation des positions des centroïdes, les *clusters* renvoyés peuvent être différents. De plus, le paramètre k ne peut pas être choisi de façon optimale à priori. C'est pourquoi, en plus d'initialiser les positions des centroïdes avec un échantillonnage en hypercubes Latins à chaque itération, la méthode ci-dessus est intégrée à une boucle pour déterminer le meilleur nombre de grappes k^* dans un domaine de valeurs prédéfini :

$$k^* \in \underset{k \in dom}{\arg\min} \sum_{i=1}^{n} \|\tilde{s}_i - centroid_k(\tilde{s}_i)\| \tag{6.16}$$

où dom définit le domaine de valeur pour k et $centroid_k(\tilde{s}_i)$ est la fonction qui retourne le centroïde de la grappe auquel $\tilde{s}_i$ appartient pour l'exécution de k-*mean* avec k *clusters*. Nous introduisons le paramètre $\mathcal{P}_K \in \{\text{Q,H}\}$ pour définir les deux domaines différents qui sont comparés dans la section des résultats numériques :

$$k \in \begin{cases} \{\lceil \frac{3}{4}\sqrt{n} \rceil, \ldots, \lfloor \sqrt{n} \rfloor\} & \text{si} \quad \mathcal{P}_K = \text{Q}, \\ \{\lceil \frac{\sqrt{n}}{2} \rceil, \ldots, \lfloor \sqrt{n} \rfloor\} & \text{si} \quad \mathcal{P}_K = \text{H}. \end{cases} \tag{6.17}$$

Finalement, les deux paramètres $\mathcal{P}_{ns}$ et $\mathcal{P}_{bbe}$ définis dans la version originelle de PSD-MADS sont étendus avec quelques différences mineures. $\mathcal{P}_{bbe}$ représente toujours le nombre maximum d'évaluations permis pour la résolution du sous-problème par MADS. Cependant, $\mathcal{P}_{ns}$ définit une borne supérieure sur le nombre de variables utilisées pour construire les sous-problèmes dans la version améliorée de PSD-MADS.

Le diagramme de la figure 6.3 résume les étapes de construction d'un sous-problème dans la version améliorée de PSD-MADS ainsi que l'endroit où chaque paramètre algorithmique intervient.

6.2 Une approche hybride

Une limitation qui encombre la méthode précédente est le fait qu'après plusieurs évaluations de la boite noire, la matrice de sensibilité tend à se stabiliser et l'algorithme k-*mean* propose les mêmes groupes de variables à chaque itération. Ceci génère des points qui ont été déjà évalués alors que les nouveaux candidats deviennent de plus en plus rares. Des tests préliminaires confirment ce phé-

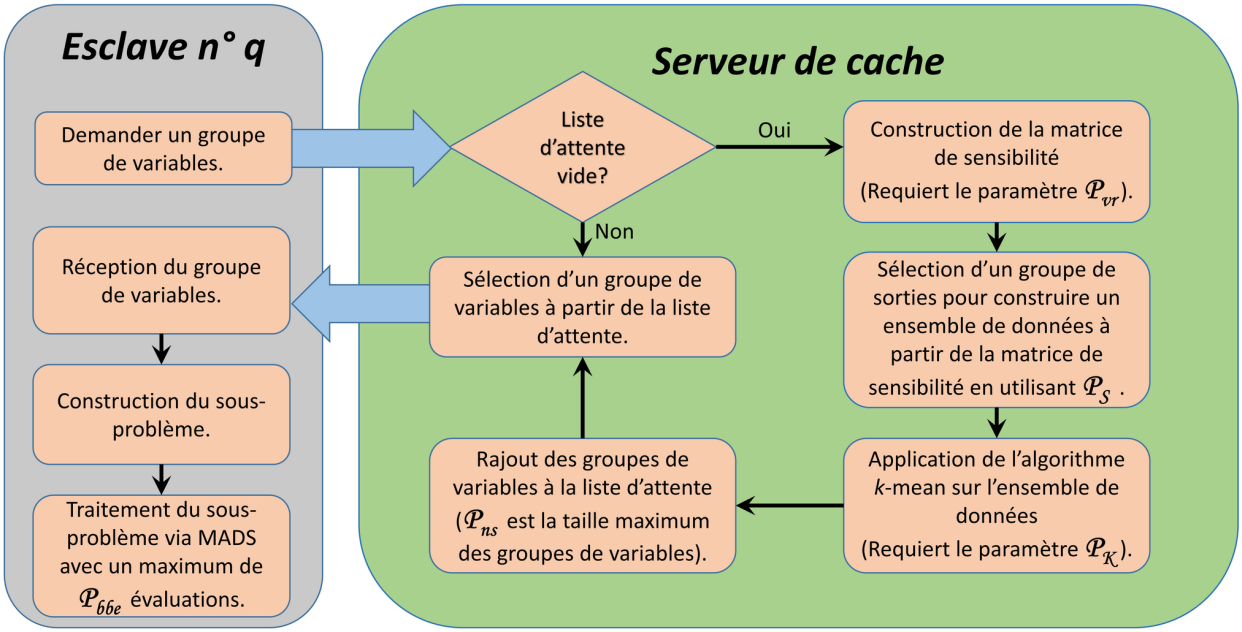


Figure 6.3 La sélection des groupes de variables pour un processus esclave.

nomène, qui devient plus accru avec l'augmentation des évaluations de boites noires. Par exemple, pour le problème B250 [30], la figure 6.4 représente le minimum, le maximum et la moyenne du nombre de succès de cache pour 30 exécutions de PSD-MADS avec la stratégie originelle et la stratégie améliorée. Nous parlons d'un succès de cache quand un élément recherché se trouve dans la cache.

L'algorithme k-*mean* remplit la file d'attente plusieurs fois avant de trouver un nouveau point, ce qui ralentit la terminaison de PSD-MADS en atteignant plus lentement le maximum nombre d'évaluations allouées.

Comme solution, nous proposons d'adopter une approche hybride qui se base sur la méthode k-*mean* précédente pour sélectionner les variables, mais change vers la sélection aléatoire utilisée dans la version originelle de PSD-MADS si la stratégie k-*mean* échoue un nombre de fois fixé à trouver un meilleur point. Après un nombre d'évaluations prédéterminé, cette approche hybride réinstaure la stratégie k-*mean* pour la sélection de variables. La sélection aléatoire sert comme une étape de diversification pour secouer les données d'entrée, modifier la matrice de sensibilité et découvrir de nouveaux groupes de variables. Pour l'exemple précèdent, la figure 6.5 reprend les résultats précèdent du problème B250 et les compare aux nombres de succès de cache de 30 exécutions de PSD-MADS avec la stratégie hybride pour la sélection de variables.

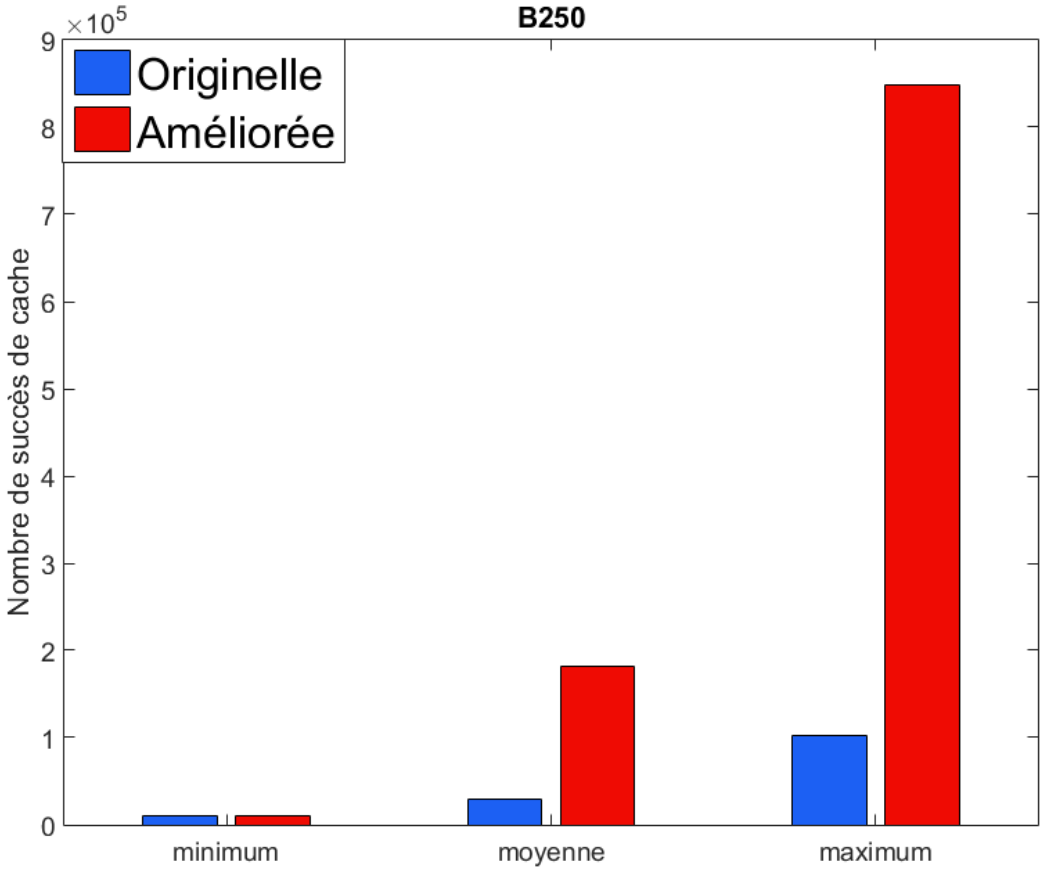


Figure 6.4 Les succès de cache pour le problème B250.

Même si, dans le pire cas, l'approche hybride semble présenter plus de succès de cache que la version améliorée, il faut remarquer que le nombre moyen de succès de cache sur les 30 instances pour la variante hybride est bien en dessous de l'approche améliorée. L'approche hybride implémentée utilise principalement l'algorithme k-*mean* pour la sélection des variables. Cependant, si cette technique échoue après trois remplissages de la liste d'attente, la version hybride change de tactique et génère $5n$ sous-problèmes via une sélection aléatoire des variables avant de faire un retour à l'approche k-*mean* de nouveau. Les paramètres utilisés pour assurer la transition entre les deux techniques de sélection de variables sont choisis pour éviter de perdre du temps à explorer les mêmes sous-espaces via k-*mean* et évaluer assez de points avec l'approche aléatoire pour perturber la matrice de sensibilité et découvrir de nouveaux groupes de variables.

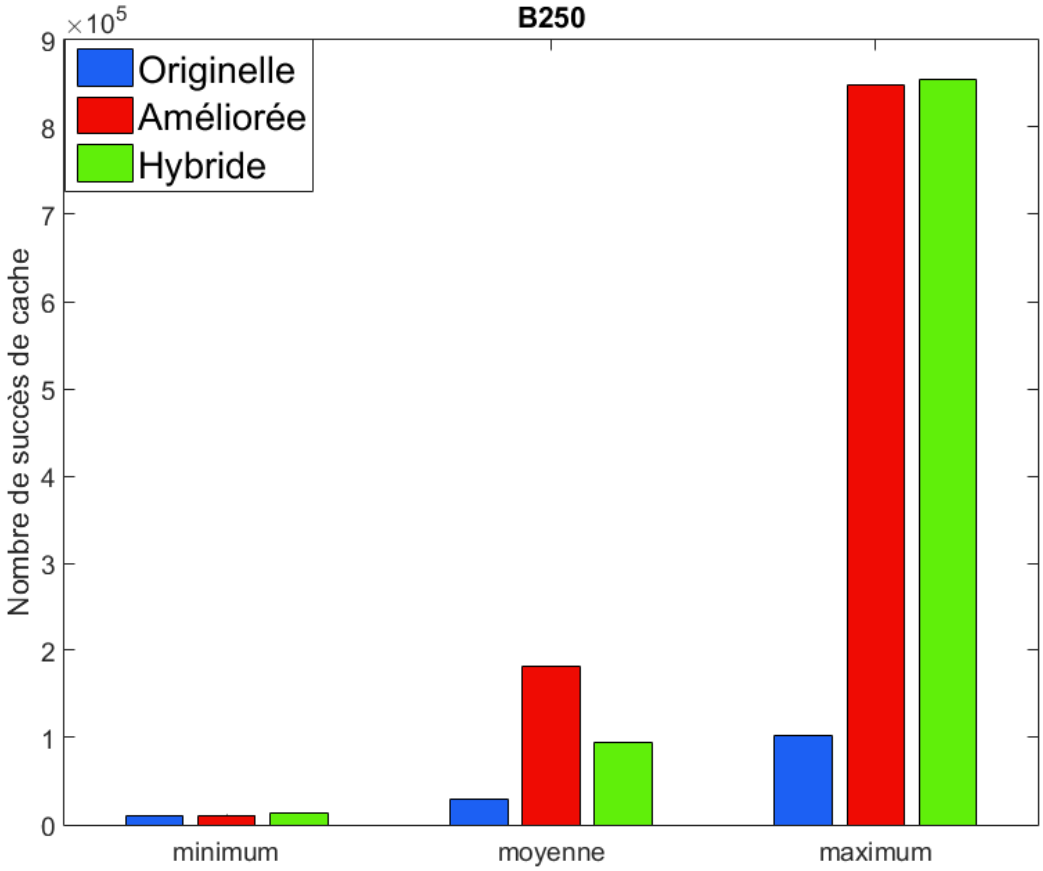


Figure 6.5 Les succès de cache pour le problème B250.

6.3 Résultats numériques

Nous comparons différentes versions de PSD-MADS sur un ensemble de problèmes contraints. Toutes les variantes de PSDMADS sont implémentées en **C++** et la parallélisation est prise en charge par **MPI**. Les tests servent à trouver des valeurs par défauts pour les paramètres définis dans la section précédente. De plus, un exemple à $4{,}000$ variables sert à comparer les versions originelle, améliorée et hybride de PSD-MADS avec les paramètres par défaut.

6.3.1 Les problèmes tests et les variantes algorithmiques

Les versions améliorée et hybride de PSD-MADS introduisent cinq nouveaux paramètres algorithmiques qui peuvent être réglés. Le tableau 6.2 résume ces paramètres ainsi que les différentes valeurs à tester pour chacun d'eux.

Tableau 6.2 Les paramètres de la nouvelle version de PSD-MADS.

Param	Description	Valeurs
$\mathcal{P}_{vr}$	Nombre d'intervalles des valeurs des variables utilisés pour construire la matrice de sensibilité.	2 : 10^2 intervalles de variables 3 : 10^3 intervalles de variables 4 : 10^4 intervalles de variables
$\mathcal{P}_K$	La gamme de valeurs utilisée pour sélectionner un bon paramètre k pour l'algorithme k-*mean*.	Q : $\lceil \frac{3}{4}\sqrt{n} \rceil \leq k \leq \lfloor \sqrt{n} \rfloor$ H : $\lceil \frac{\sqrt{n}}{2} \rceil \leq k \leq \lfloor \sqrt{n} \rfloor$
$\mathcal{P}_S$	La stratégie de groupement de sorties pour utiliser k-*mean* sur une sous partie de la matrice de sensibilité.	S1 : La matrice de sensibilité en entier S2 : L'objectif d'abord puis toutes les contraintes S3 : L'objectif d'abord puis la matrice de sensibilité S4 : Sélection de deux sorties aléatoirement
$\mathcal{P}_{ns}$	Borne supérieure sur le nombre de variables utilisées pour construire les sous-problèmes.	2 : La dimension maximale des sous-problèmes est 2 3 : La dimension maximale des sous-problèmes est 3 5 : La dimension maximale des sous-problèmes est 5 10 : La dimension maximale des sous-problèmes est 10
$\mathcal{P}_{bbe}$	Nombre maximum d'évaluations de la boite noire permis pour chaque sous-problème.	10 : 10 évaluations 20 : 20 évaluations 30 : 30 évaluations

Comme nous l'avons vu précédemment, la version originelle de PSD-MADS sélectionne aléatoirement $\mathcal{P}_{ns}$ variables pour construire un sous-problème avec un budget de $\mathcal{P}_{bbe}$ évaluations. Pour les tests menés dans la suite de la section, la version originelle est dénotée Originelle-$\mathcal{P}_{ns}$-$\mathcal{P}_{bbe}$. Pour les méthodes améliorées qui utilisent l'algorithme k-*mean* et pour lesquelles tous les paramètres sont définis, la notation suivante est adoptée : Améliorée-$\mathcal{P}_{ns}$-$\mathcal{P}_{bbe}$-$\mathcal{P}_{vr}$ $\mathcal{P}_K$ $\mathcal{P}_S$. Le tableau 6.3 présente toutes les versions de PSD-MADS testées lors de la phase de sélection de valeurs par défaut.

Tableau 6.3 Liste des algorithmes utilisés pour fixer les valeur par défaut des paramètres.

Algorithme	$\mathcal{P}_{ns}$	$\mathcal{P}_{bbe}$	$\mathcal{P}_{vr}$	$\mathcal{P}_K$	$\mathcal{P}_S$	Algorithme	$\mathcal{P}_{ns}$	$\mathcal{P}_{bbe}$	$\mathcal{P}_{vr}$	$\mathcal{P}_K$	$\mathcal{P}_S$
Améliorée-5-20-3QS1	5	20	3	Q	S1	Améliorée-2-20-3QS1	2	20	3	Q	S1
Améliorée-5-20-3QS2	5	20	3	Q	S2	Améliorée-3-20-3QS1	3	20	3	Q	S1
Améliorée-5-20-3QS3	5	20	3	Q	S3	Améliorée-10-20-3QS1	10	20	3	Q	S1
Améliorée-5-20-3QS4	5	20	3	Q	S4	Originelle-5-10	5	10			
Améliorée-5-20-2QS1	5	20	2	Q	S1	Originelle-5-20	5	20			
Améliorée-5-20-4QS1	5	20	4	Q	S1	Originelle-5-30	5	30			
Améliorée-5-20-3HS1	5	20	3	H	S1	Originelle-2-20	2	20			
Améliorée-5-10-3QS1	5	10	3	Q	S1	Originelle-3-20	3	20			
Améliorée-5-30-3QS1	5	30	3	Q	S1	Originelle-10-20	10	20			

Le tableau 6.4 liste les problèmes d'optimisation contraints utilisés pour tester nos variantes algorithmiques. Les problèmes tests ont des dimensions entre de 60 à $4{,}000$ variables.

Pour chaque problème test de dimension n, l'exécution d'une des variantes algorithmiques est permise avec un maximum de $100n$ évaluations de la boite noire, avec une exception pour les problèmes B250 et B500 qui permettent un maximum de $60{,}000$ évaluations. Les versions améliorée et hybride utilisent des aspects aléatoires pour la sélection de variables, l'initialisation des centroïdes

Tableau 6.4 Liste des problèmes tests.

nom	n	m	source	nom	n	m	source	nom	n	m	source
B250	60	1	[30]	CRESCENT-2000	2,000	2	[19]	G2-500	500	2	[126]
B500	60	1	[30]	DISK-500	500	1	[19]	G2-1000	1,000	2	[126]
CRESCENT-500	500	2	[19]	DISK-1000	1,000	1	[19]	G2-2000	2,000	2	[126]
CRESCENT-1000	1,000	2	[19]	DISK-2000	2,000	1	[19]	G2-4000	4,000	2	[126]

pour *k-mean* ou encore pour la sélection de sorties via la stratégie S4. Pour ceci, l'exécution d'une variante algorithmique sur un problème est effectuée 30 fois à partir du même point de départ. Ces exécutions sont menées en parallèle en utilisant 500 processeurs de CASIR, le superordinateur d'Hydro-Québec avec 4300 coeurs et 17TB de mémoire vive distribuée.

En plus de profils de données, nous utilisons des tableaux de résultats pour analyser les comparaisons entre les différentes variantes. Pour chaque problème et pour chaque algorithme, le tableau de résultats donne la meilleure (Z_{best}) et la pire (Z_{worst}) valeurs de la fonction objectif parmi les solutions des 30 instances exécutées. De plus, le tableau reporte la valeur moyenne (Z_{avg}) des solutions des 30 instances. Les meilleures valeurs sont en gras pour chaque catégorie.

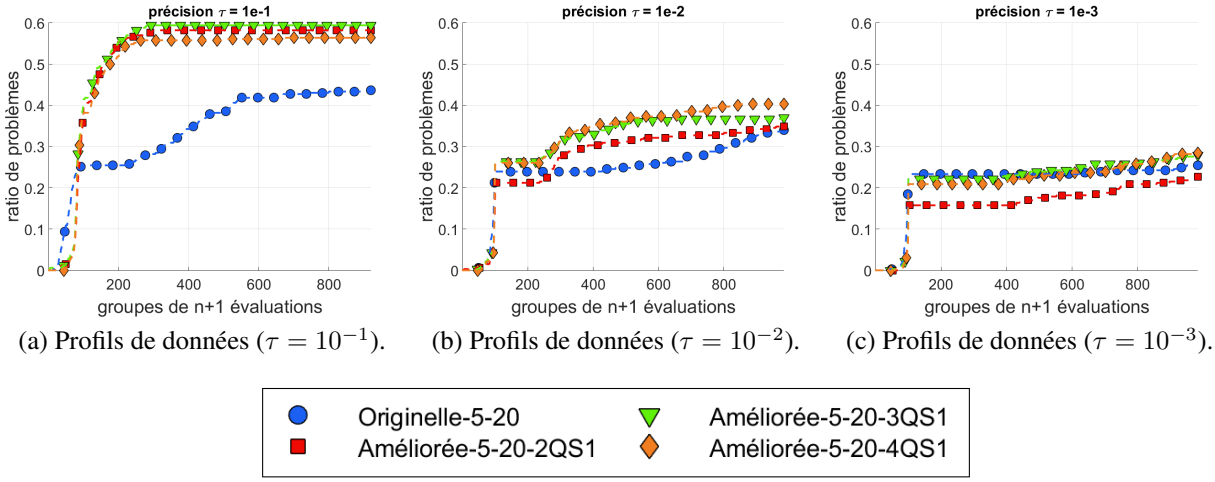
6.3.2 Fixation des valeurs par défaut des paramètres de l'algorithme amélioré

Nous commençons toujours par fixer les paramètres algorithmiques aux valeurs suivantes : $\mathcal{P}_{vr} = 3$, $\mathcal{P}_K =$Q, $\mathcal{P}_S =$S1, $\mathcal{P}_{ns} = 5$ et $\mathcal{P}_{bbe} = 20$. Cette section est divisée en cinq parties : une pour chaque paramètre. Dans chaque partie, tous les paramètres sont fixés aux valeurs citées ci-dessus sauf pour le paramètre étudié qui prend toutes les valeurs correspondantes citées au tableau 6.2. Le but de ces tests est de déterminer un jeu de valeurs à intégrer dans NOMAD comme des valeurs par défaut pour les paramètres algorithmiques des versions améliorée et hybride de PSD-MADS. Nous utilisons tous les problèmes du tableau 6.4 sauf le problème G2-4000 que nous traitons séparément à la fin de cette section. Cette séparation est établie pour pouvoir exécuter tous les tests dans le temps qui nous est imparti pour l'utilisation de CASIR et limiter l'optimisation de G2-4000 qui est très coûteuse temporellement.

Paramètre $\mathcal{P}_{vr}$

Le paramètre $\mathcal{P}_{vr}$, nécessaire pour la construction de la matrice de sensibilité, est testé avec les valeurs 2, 3 et 4. La figure 6.6 et le tableau 6.5 comparent les stratégies Originelle-5-20, Améliorée-5-20-2QS1, Améliorée-5-20-3QS1 et Améliorée-5-20-4QS1.

Comme défini dans le chapitre 4, un profil de données donne une représentation graphique du ratio de problèmes résolus par un algorithme donné pour une tolérance τ et pour un nombre déterminé de



(a) Profils de données ($\tau = 10^{-1}$). (b) Profils de données ($\tau = 10^{-2}$). (c) Profils de données ($\tau = 10^{-3}$).

Figure 6.6 Comparaison des méthodes originelle et améliorées avec différents nombres de gammes de variables $10^{\mathcal{P}_{vr}}$ pour la construction de la matrice de sensibilité.

Tableau 6.5 Sommaire des résultats pour la sélection d'une valeur par défaut de $\mathcal{P}_{vr}$.

Algo.	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-5-20	G2-500	-0.15432	-0.058469	-0.11804	CRESCENT-500	-40.52	-12.61	-19.453
Améliorée-5-20-2QS1		-0.26004	-0.22505	-0.2374		-54.516	-11.61	-25.039
Améliorée-5-20-3QS1		**-0.26114**	-0.22541	-0.23881		**-55.54**	-17.01	**-27.401**
Améliorée-5-20-4QS1		-0.26001	**-0.22741**	**-0.23909**		-42.48	**-17.21**	-25.227
Originelle-5-20	G2-1000	-0.13701	-0.062465	-0.096506	CRESCENT-1000	-66.823	-17.37	-38.851
Améliorée-5-20-2QS1		-0.25953	-0.23103	-0.24476		-145.7	**-61.61**	**-93.59**
Améliorée-5-20-3QS1		-0.2581	-0.2331	**-0.24885**		**-171.971**	-44.524	-88.469
Améliorée-5-20-4QS1		**-0.26058**	**-0.23444**	-0.24744		-153.991	-17.669	-86.298
Originelle-5-20	G2-2000	-0.10528	-0.066553	-0.086113	CRESCENT-2000	-120.354	-31.78	-64.303
Améliorée-5-20-2QS1		-0.24079	-0.21676	-0.23287		-208.05	-72.917	-136.706
Améliorée-5-20-3QS1		-0.24533	**-0.22492**	-0.23985		-199.131	**-84.175**	**-137.266**
Améliorée-5-20-4QS1		**-0.24694**	-0.21825	**-0.23987**		**-223.023**	-64.933	-131.714
Originelle-5-20	DISK-500	**-34.73**	**-29.1**	**-31.415**	B250	9.754	120.996	31.894
Améliorée-5-20-2QS1		-26.435	-15.425	-21.231		9.3423	**45.956**	21.161
Améliorée-5-20-3QS1		-28.482	-15.02	-22.027		**8.6219**	53.388	**19.429**
Améliorée-5-20-4QS1		-30.284	-14.193	-20.782		9.4575	54.247	21.361
Originelle-5-20	DISK-1000	-36.13	**-29.85**	**-33.797**	B500	130.334	342.251	258.683
Améliorée-5-20-2QS1		-49.34	-15.479	-27.285		**74.512**	283.198	211.552
Améliorée-5-20-3QS1		-52.44	-16.298	-26.921		119.211	399.753	234.291
Améliorée-5-20-4QS1		**-55.78**	-16.29	-25.411		81.064	**282.989**	**205.239**
Originelle-5-20	DISK-2000	**-31.96**	**-30**	**-30.95**				
Améliorée-5-20-2QS1		-30.532	-15.026	-22.33				
Améliorée-5-20-3QS1		-30.783	-14.956	-21.954				
Améliorée-5-20-4QS1		-30.718	-14.518	-21.264				

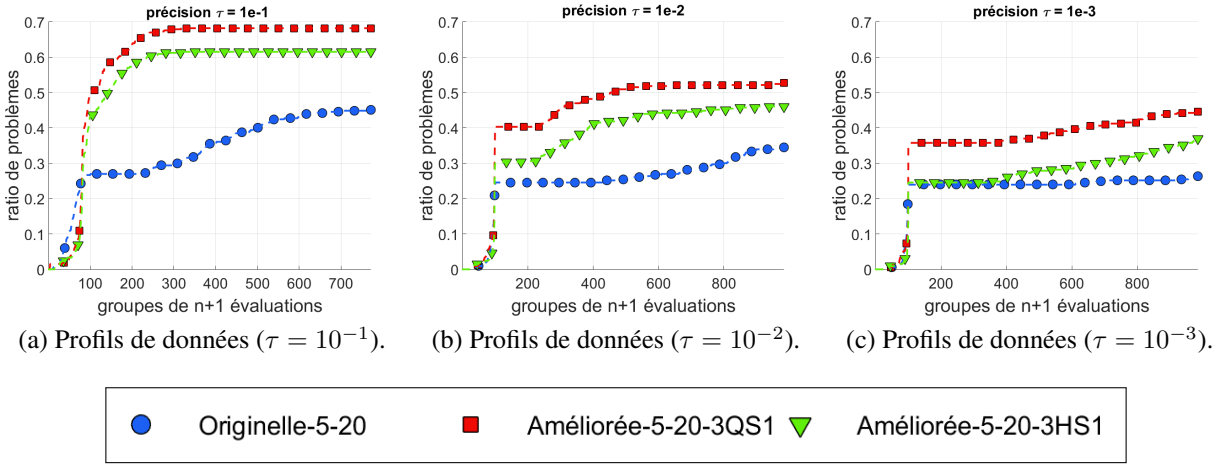
groupes de $n+1$ évaluations. La comparaison entre les profils de données des différents algorithmes détermine la meilleure méthode pour un budget fixé.

Il est difficile de dégager un gagnant parmi les trois variantes améliorées depuis les profils de données de la figure 6.6. Cependant, pour les précisions $\tau = 10^{-2}$ et $\tau = 10^{-3}$, Améliorée-5-20-

4QS1 surpassent le reste des méthodes après $900(n+1)$ évaluations. Ceci s'accorde avec le fait que plus la valeur de $\mathcal{P}_{vr}$ est grande, plus la matrice de sensibilité est précise.

Paramètre $\mathcal{P}_K$

Le paramètre $\mathcal{P}_K$, qui permet de sélectionner une bonne valeur k pour l'algorithme k-*mean*, prend soit la valeur Q ou la valeur H. La figure 6.7 et le tableau 6.6 résument la comparaison entre Originelle-5-20, Améliorée-5-20-3QS1 et Améliorée-5-20-3HS1.



(a) Profils de données ($\tau = 10^{-1}$). (b) Profils de données ($\tau = 10^{-2}$). (c) Profils de données ($\tau = 10^{-3}$).

Figure 6.7 Comparaison des méthodes améliorées avec différents intervalles du paramètre k pour l'algorithme k-*mean* ($\mathcal{P}_K$).

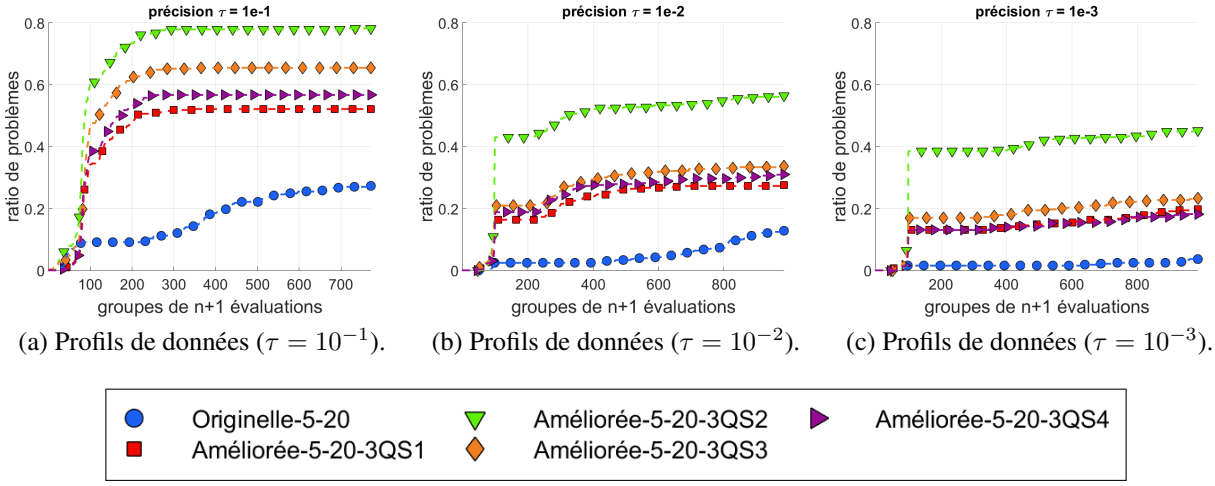
Tableau 6.6 Sommaire des résultats pour la sélection d'une valeur par défaut de $\mathcal{P}_K$.

Algo.	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-5-20	G2-500	-0.15432	-0.058469	-0.11804	CRESCENT-500	-40.52	-12.61	-19.453
Améliorée-5-20-3QS1		**-0.26114**	-0.22541	**-0.23881**		**-55.54**	**-17.01**	**-27.401**
Améliorée-5-20-3HS1		-0.25583	**-0.22621**	-0.23443		-47.024	-11.78	-25.584
Originelle-5-20	G2-1000	-0.13701	-0.062465	-0.096506	CRESCENT-1000	-66.823	-17.37	-38.851
Améliorée-5-20-3QS1		-0.2581	-0.2331	**-0.24885**		**-171.971**	-44.524	**-88.469**
Améliorée-5-20-3HS1		**-0.26257**	**-0.23435**	-0.2453		-119.947	**-52.061**	-82.981
Originelle-5-20	G2-2000	-0.10528	-0.066553	-0.086113	CRESCENT-2000	-120.354	-31.78	-64.303
Améliorée-5-20-3QS1		**-0.24533**	-0.22492	-0.23985		-199.131	-84.175	**-137.266**
Améliorée-5-20-3HS1		-0.24479	**-0.23771**	**-0.24068**		**-200.206**	**-86.364**	-119.8
Originelle-5-20	DISK-500	**-34.73**	**-29.1**	**-31.415**	B250	9.754	120.996	31.894
Améliorée-5-20-3QS1		-28.482	-15.02	-22.027		8.6219	53.388	19.429
Améliorée-5-20-3HS1		-26.108	-16	-21.707		**8.4904**	**36.778**	**17.383**
Originelle-5-20	DISK-1000	-36.13	**-29.85**	**-33.797**	B500	130.334	342.251	258.683
Améliorée-5-20-3QS1		**-52.44**	-16.298	-26.921		119.211	399.753	234.291
Améliorée-5-20-3HS1		-48.81	-14.645	-27.309		**85.71**	**295.272**	**186.026**
Originelle-5-20	DISK-2000	-31.96	**-30**	**-30.95**				
Améliorée-5-20-3QS1		-30.783	-14.956	-21.954				
Améliorée-5-20-3HS1		**-34.393**	-14.787	-22.713				

Les deux variantes améliorées de PSD-MADS sont meilleures que la version originelle. Même si le paramètre $\mathcal{P}_K$ =H couvre une plus grande gamme de valeurs de k (pour la méthode k-*mean*) que $\mathcal{P}_K$ =Q, les résultats suggèrent qu'adopter $\mathcal{P}_K$ =Q donne de bien meilleurs résultats. La figure 6.7 montre que la méthode Améliorée-5-20-3QS1 est légèrement supérieure.

Paramètre $\mathcal{P}_S$

Le paramètre $\mathcal{P}_S$, qui représente la stratégie de sélection des sorties pour mieux exploiter la matrice de sensibilité, va désigner une des quatre stratégies S1, S2, S3 et S4 définies précédemment. La figure 6.8 et le tableau 6.7 résument la comparaison entre Originelle-5-20, Améliorée-5-20-3QS1, Améliorée-5-20-3QS2, Améliorée-5-20-3QS3 et Améliorée-5-20-3QS4.



(a) Profils de données ($\tau = 10^{-1}$). (b) Profils de données ($\tau = 10^{-2}$). (c) Profils de données ($\tau = 10^{-3}$).

Figure 6.8 Comparaison des méthodes avec différentes stratégies $\mathcal{P}_{vr}$ pour regrouper les sorties.

L'algorithme Améliorée-5-20-3QS2 remporte largement cette manche. Les stratégies S2 et S3 exécutent deux fois la méthode de classification au lieu d'une seule fois, ce qui peut expliquer leurs résultats supérieurs comparés aux deux autres stratégies. Il y a aussi un grand écart entre les stratégies S2 et S3. Nous pouvons expliquer ceci par le fait que la deuxième phase de classification de la stratégie S2 s'intéresse uniquement aux contraintes alors que la même phase dans S3 applique la classification sur la matrice de sensibilité entière. Se concentrer uniquement sur les contraintes pourrait conduire à la construction de sous-problèmes qui sont plus aptes à générer des candidats réalisables et qui permettent une meilleure exploration de l'espace des solutions. À noter que toutes les versions améliorées de PSD-MADS donnent de meilleurs résultats que la version originelle de PSD-MADS. Dans le tableau 6.7, la version améliorée de PSD-MADS utilisant la stratégie de groupement de sorties S2 trouve la meilleure solution moyenne pour 7 problèmes sur 11, alors que l'approche Originelle-5-20 échoue à trouver la meilleure solution pour tous les problèmes.

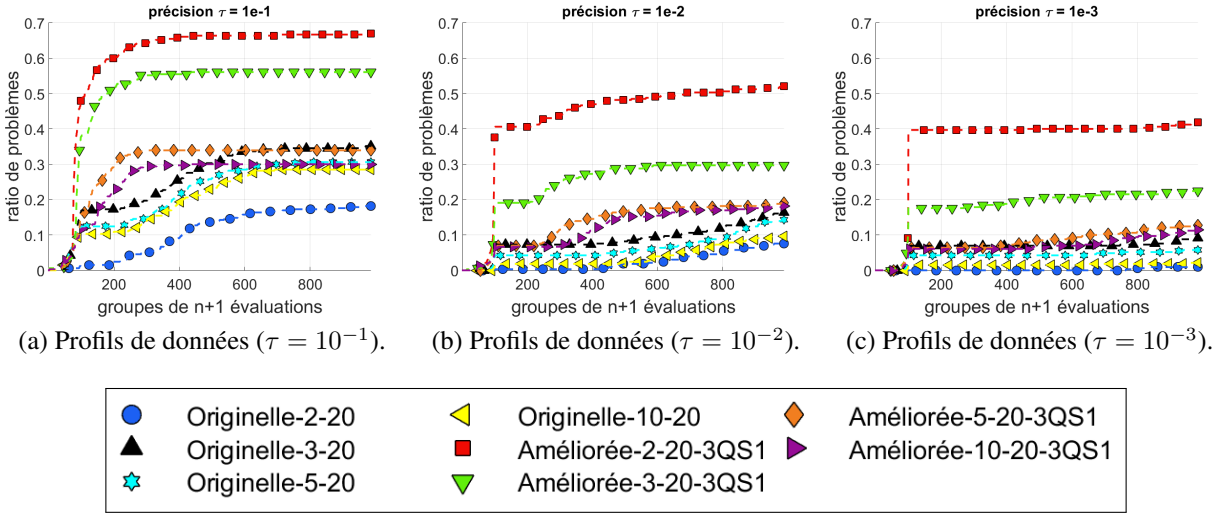
Tableau 6.7 Sommaire des résultats pour la sélection d'une valeur par défaut de $\mathcal{P}_S$.

Algo.	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-5-20	G2-500	-0.15432	-0.058469	-0.11804	CRESCENT-500	-40.52	-12.61	-19.453
Améliorée-5-20-3QS1		-0.26114	-0.22541	-0.23881		**-55.54**	**-17.01**	**-27.401**
Améliorée-5-20-3QS2		-0.26597	**-0.23356**	**-0.2432**		-41.46	-14.99	-22.719
Améliorée-5-20-3QS3		**-0.26709**	-0.22717	-0.24069		-43.32	-11.8	-27.175
Améliorée-5-20-3QS4		-0.26062	-0.22867	-0.24235		-51.883	-15.14	-25.479
Originelle-5-20	G2-1000	-0.13701	-0.062465	-0.096506	CRESCENT-1000	-66.823	-17.37	-38.851
Améliorée-5-20-3QS1		-0.2581	-0.2331	-0.24885		**-171.971**	-44.524	**-88.469**
Améliorée-5-20-3QS2		**-0.26277**	-0.23968	**-0.2519**		-121.6	-59.541	-79.613
Améliorée-5-20-3QS3		-0.25846	-0.23621	-0.24865		-126.97	**-63.837**	-85.418
Améliorée-5-20-3QS4		-0.26259	**-0.24106**	-0.251		-130.965	-27.129	-82.672
Originelle-5-20	G2-2000	-0.10528	-0.066553	-0.086113	CRESCENT-2000	-120.354	-31.78	-64.303
Améliorée-5-20-3QS1		-0.24533	-0.22492	-0.23985		-199.131	-84.175	-137.266
Améliorée-5-20-3QS2		-0.24648	-0.22569	-0.24221		-190.302	**-95.112**	-135.266
Améliorée-5-20-3QS3		-0.24662	-0.23537	-0.24032		**-222.991**	-78.327	**-141.418**
Améliorée-5-20-3QS4		**-0.24754**	**-0.23819**	**-0.24257**		-210.897	-86.309	-135.36
Originelle-5-20	DISK-500	-34.73	-29.1	-31.415	B250	9.754	120.996	31.894
Améliorée-5-20-3QS1		-28.482	-15.02	-22.027		**8.6219**	53.388	19.429
Améliorée-5-20-3QS2		**-52.77**	**-41.14**	**-48.092**		8.674	**52.844**	**18.114**
Améliorée-5-20-3QS3		-47.34	-14.549	-32.65		9.1533	55.458	20.031
Améliorée-5-20-3QS4		-37.69	-14.98	-22.315		9.5394	84.301	21.126
Originelle-5-20	DISK-1000	-36.13	-29.85	-33.797	B500	130.334	342.251	258.683
Améliorée-5-20-3QS1		-52.44	-16.298	-26.921		**119.211**	399.753	234.291
Améliorée-5-20-3QS2		**-68.43**	**-56.64**	**-62.782**		119.994	**326.975**	**219.381**
Améliorée-5-20-3QS3		-63.9	-26.75	-53.891		121.964	350.398	223.567
Améliorée-5-20-3QS4		-48.623	-18.241	-26.855		122.078	377.969	239.476
Originelle-5-20	DISK-2000	-31.96	-30	-30.95				
Améliorée-5-20-3QS1		-30.783	-14.956	-21.954				
Améliorée-5-20-3QS2		-34.404	**-30.846**	**-32.212**				
Améliorée-5-20-3QS3		**-35.221**	-20.736	-28.429				
Améliorée-5-20-3QS4		-31.575	-16.634	-23.792				

Paramètre $\mathcal{P}_{ns}$

Le paramètre $\mathcal{P}_{ns}$ représente soit la taille des sous-problèmes dans la version originelle, soit la borne supérieure de la taille des sous-problèmes pour la version améliorée. Le paramètre varie entre les valeurs 2, 3, 5 et 10 dans cette partie. La figure 6.9 et le tableau 6.8 comparent entre Originelle-2-20, Originelle-3-20, Originelle-5-20, Originelle-10-20, Améliorée-2-20-3QS1, Améliorée-3-20-3QS1, Améliorée-5-20-3QS1 et Améliorée-10-20-3QS1.

La version améliorée surpasse encore une fois la version originelle de PSD-MADS. Les variantes originelles échouent à trouver la meilleure solution pour tous les problèmes tests selon le tableau 6.8. Les profils de données de la figure 6.9 montrent qu'utiliser deux variables comme borne supérieure sur la taille des sous-problèmes donne de meilleurs résultats que le reste des versions. Il paraît que traiter des sous-problèmes de très petite dimension permet à PSD-MADS de mieux explorer l'espace des solutions. De plus, MADS est beaucoup plus efficace avec des sous-problèmes de cette taille.

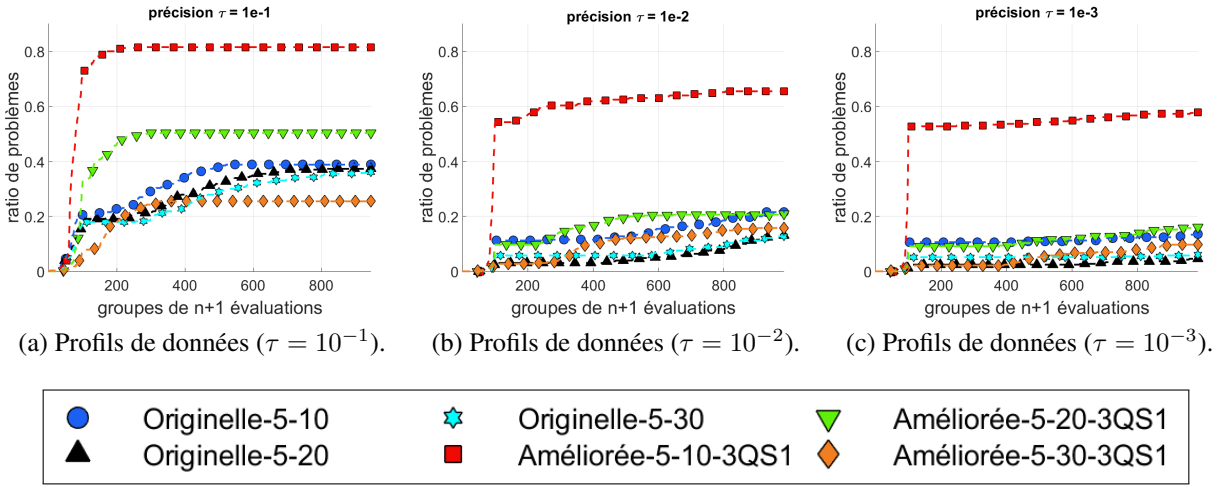


(a) Profils de données ($\tau = 10^{-1}$). (b) Profils de données ($\tau = 10^{-2}$). (c) Profils de données ($\tau = 10^{-3}$).

Figure 6.9 Sélection d'une valeur par défaut de la borne supérieure de la taille des sous-problèmes $\mathcal{P}_{ns}$ pour la version améliorée de PSD-MADS.

Paramètre $\mathcal{P}_{bbe}$

Le paramètre $\mathcal{P}_{bbe}$ représente le nombre d'évaluations de boites noires alloué pour le traitement de chaque sous-problème. Dans cette partie des tests, $\mathcal{P}_{bbe}$ varie entre 10, 20 et 30. La figure 6.10 et le tableau 6.9 résument la comparaison entre Originelle-5-10, Originelle-5-20, Originelle-5-30, Améliorée-5-10-3QS1, Améliorée-5-20-3QS1 et Améliorée-5-30-3QS1.



(a) Profils de données ($\tau = 10^{-1}$). (b) Profils de données ($\tau = 10^{-2}$). (c) Profils de données ($\tau = 10^{-3}$).

Figure 6.10 Sélection d'une valeur par défaut pour la valeur du nombre maximum d'évaluations $\mathcal{P}_{bbe}$ pour les sous-problèmes.

Il semble que la version améliorée avec un nombre d'évaluations $\mathcal{P}_{bbe} = 10$ pour chaque sous-

Tableau 6.8 Sommaire des résultats pour la sélection d'une valeur par défaut de $\mathcal{P}_{ns}$.

Algo.	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-2-20	G2-500	-0.14577	-0.052585	-0.091178	CRESCENT-500	-39.13	**-19.56**	-21.899
Originelle-3-20		-0.14485	-0.06184	-0.11946		-38.95	-13.1	-20.941
Originelle-5-20		-0.15432	-0.058469	-0.11804		-40.52	-12.61	-19.453
Originelle-10-20		-0.14323	-0.067681	-0.12505		-36.31	-11.47	-18.35
Améliorée-2-20-3QS1		**-0.31547**	**-0.2513**	**-0.27636**		-50.14	-14.58	**-31.346**
Améliorée-3-20-3QS1		-0.28944	-0.24374	-0.25938		**-68.448**	-14.24	-28.904
Améliorée-5-20-3QS1		-0.26114	-0.22541	-0.23881		-55.54	-17.01	-27.401
Améliorée-10-20-3QS1		-0.22417	-0.19534	-0.20544		-44.57	-16.5	-26.723
Originelle-2-20	G2-1000	-0.10528	-0.047408	-0.07147	CRESCENT-1000	-87.14	-26.83	-51.846
Originelle-3-20		-0.10699	-0.053706	-0.090355		-110.19	-26.18	-51.414
Originelle-5-20		-0.13701	-0.062465	-0.096506		-66.823	-17.37	-38.851
Originelle-10-20		-0.13423	-0.068917	-0.1067		-93.04	-15.22	-42.722
Améliorée-2-20-3QS1		**-0.31005**	**-0.25988**	**-0.28741**		**-182.023**	-37.786	-86.811
Améliorée-3-20-3QS1		-0.28849	-0.25486	-0.27264		-181.829	**-64.903**	**-98.764**
Améliorée-5-20-3QS1		-0.2581	-0.2331	-0.24885		-171.971	-44.524	-88.469
Améliorée-10-20-3QS1		-0.2207	-0.20107	-0.2124		-133.803	-28.79	-78.2
Originelle-2-20	G2-2000	-0.09804	-0.046032	-0.067732	CRESCENT-2000	-104.707	-9.999	-54.977
Originelle-3-20		-0.10168	-0.059882	-0.078319		-140.671	-38.623	-86.686
Originelle-5-20		-0.10528	-0.066553	-0.086113		-120.354	-31.78	-64.303
Originelle-10-20		-0.11127	-0.076395	-0.10166		-135.626	-63.178	-97.029
Améliorée-2-20-3QS1		**-0.28235**	**-0.26579**	**-0.27223**		-182.981	-73.157	-121.76
Améliorée-3-20-3QS1		-0.26603	-0.23393	-0.25874		**-253.428**	-63.209	-131.263
Améliorée-5-20-3QS1		-0.24533	-0.22492	-0.23985		-199.131	-84.175	**-137.266**
Améliorée-10-20-3QS1		-0.21908	-0.20409	-0.21606		-208.518	**-90.307**	-129
Originelle-2-20	DISK-500	-28.37	-19.03	-24.161	B250	10.155	97.695	40.677
Originelle-3-20		-33.89	-27.76	-31.279		12.005	183.113	41.422
Originelle-5-20		-34.73	**-29.1**	**-31.415**		9.754	120.996	31.894
Originelle-10-20		-32.87	-27.57	-30.896		10.487	59.545	35.535
Améliorée-2-20-3QS1		-35.691	-15.84	-22.469		9.0427	99.181	28.686
Améliorée-3-20-3QS1		**-41.78**	-14.553	-22.784		8.8202	**48.362**	19.434
Améliorée-5-20-3QS1		-28.482	-15.02	-22.027		**8.6219**	53.388	**19.429**
Améliorée-10-20-3QS1		-30.203	-17.18	-21.864		9.0203	109.159	26.267
Originelle-2-20	DISK-1000	-33.8	-25.23	-29.264	B500	219.455	604.108	332.658
Originelle-3-20		-37.51	-30.32	-34.068		139.685	428.253	257.58
Originelle-5-20		-36.13	-29.85	-33.797		130.334	**342.251**	258.683
Originelle-10-20		-36.05	**-30.84**	-33.399		199.813	402.831	280.528
Améliorée-2-20-3QS1		-47.62	-18.93	**-35.297**		135.578	455.519	237.859
Améliorée-3-20-3QS1		**-54.063**	-18.315	-32.294		133.714	384.008	249.2
Améliorée-5-20-3QS1		-52.44	-16.298	-26.921		**119.211**	399.753	**234.291**
Améliorée-10-20-3QS1		-39.35	-19.824	-27.178		120.905	473.428	271.986
Originelle-2-20	DISK-2000	-32.19	-25.67	-28.108				
Originelle-3-20		-35.93	**-32.09**	-33.326				
Originelle-5-20		-31.96	-30	-30.95				
Originelle-10-20		-30.486	-30	-30.034				
Améliorée-2-20-3QS1		**-44.38**	-19.989	**-34.637**				
Améliorée-3-20-3QS1		-35.728	-16.612	-24.87				
Améliorée-5-20-3QS1		-30.783	-14.956	-21.954				
Améliorée-10-20-3QS1		-32.086	-16.48	-23.229				

problème surpasse toutes les autres versions avec un très grand écart. Dans la figure 6.10(b), la méthode Améliorée-5-10-3QS1 résout quasiment 65% des instances, alors que les autres méthodes sont au-dessous des 20% après $900(n+1)$ évaluations. Quand le nombre d'évaluations allouées est petit, PSD-MADS construit plus de sous-problèmes et, par conséquent, assure une meilleure couverture de l'espace des solutions.

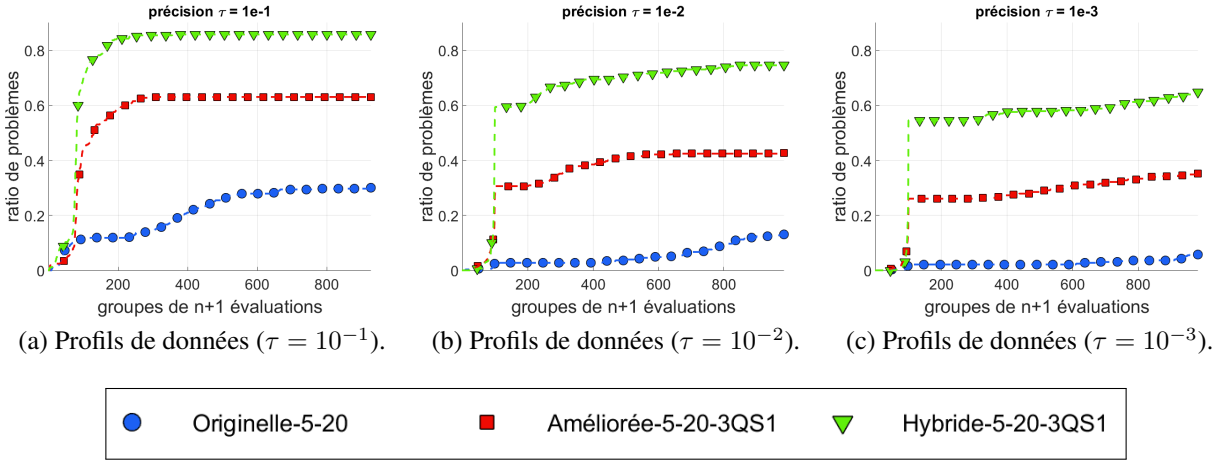
Tableau 6.9 Sommaire des résultats pour la sélection d'une valeur par défaut de $\mathcal{P}_{bbe}$.

Algo.	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-5-10	G2-500	-0.1664	-0.082659	-0.13267	CRESCENT-500	-36.051	-13.68	-19.551
Originelle-5-20		-0.15432	-0.058469	-0.11804		-40.52	-12.61	-19.453
Originelle-5-30		-0.14215	-0.049323	-0.11446		-41.04	-14.36	-20.104
Améliorée-5-10-3QS1		**-0.27789**	**-0.23689**	**-0.25427**		**-80.18**	**-17.66**	**-41.442**
Améliorée-5-20-3QS1		-0.26114	-0.22541	-0.23881		-55.54	-17.01	-27.401
Améliorée-5-30-3QS1		-0.243	-0.20259	-0.21673		-44.72	-6.22	-23.297
Originelle-5-10	G2-1000	-0.1204	-0.071093	-0.089195	CRESCENT-1000	-84.05	-21.29	-48.709
Originelle-5-20		-0.13701	-0.062465	-0.096506		-66.823	-17.37	-38.851
Originelle-5-30		-0.13319	-0.052187	-0.092342		-70.48	-20.96	-41.627
Améliorée-5-10-3QS1		**-0.27525**	**-0.24795**	**-0.26334**		**-180.677**	**-91.694**	**-119.225**
Améliorée-5-20-3QS1		-0.2581	-0.2331	-0.24885		-171.971	-44.524	-88.469
Améliorée-5-30-3QS1		-0.23569	-0.20906	-0.22348		-123.475	-27.65	-60.794
Originelle-5-10	G2-2000	-0.11613	-0.078508	-0.096499	CRESCENT-2000	-144.046	-44.535	-85.594
Originelle-5-20		-0.10528	-0.066553	-0.086113		-120.354	-31.78	-64.303
Originelle-5-30		-0.10129	-0.061909	-0.084825		-139.688	-18.12	-74.927
Améliorée-5-10-3QS1		**-0.26852**	**-0.25439**	**-0.26099**		**-230.747**	**-113.516**	**-164.447**
Améliorée-5-20-3QS1		-0.24533	-0.22492	-0.23985		-199.131	-84.175	-137.266
Améliorée-5-30-3QS1		-0.21081	-0.18357	-0.2064		-174.565	-53.759	-117.628
Originelle-5-10	DISK-500	**-36.03**	**-31.11**	**-33.821**	B250	8.9707	93.814	28.668
Originelle-5-20		-34.73	-29.1	-31.415		9.754	120.996	31.894
Originelle-5-30		-32.97	-26.45	-30.526		12.027	188.872	48.506
Améliorée-5-10-3QS1		-33.025	-15.401	-23.324		**8.502**	**47.396**	22.329
Améliorée-5-20-3QS1		-28.482	-15.02	-22.027		8.6219	53.388	**19.429**
Améliorée-5-30-3QS1		-29.06	-13.101	-21.887		8.7733	56.66	19.543
Originelle-5-10	DISK-1000	-40.35	**-32.45**	**-35.351**	B500	142.414	**296.104**	232.947
Originelle-5-20		-36.13	-29.85	-33.797		130.334	342.251	258.683
Originelle-5-30		-36.85	-31.45	-33.729		154.782	402.603	265.851
Améliorée-5-10-3QS1		-48.622	-21.379	-34.925		87.915	392.83	220.743
Améliorée-5-20-3QS1		**-52.44**	-16.298	-26.921		119.211	399.753	234.291
Améliorée-5-30-3QS1		-37.581	-15.991	-23.581		**80.101**	381.137	**209.849**
Originelle-5-10	DISK-2000	-30.436	-30	-30.027				
Originelle-5-20		-31.96	-30	-30.95				
Originelle-5-30		-33.65	**-30.72**	-32.15				
Améliorée-5-10-3QS1		**-37.09**	-23.119	**-32.225**				
Améliorée-5-20-3QS1		-30.783	-14.956	-21.954				
Améliorée-5-30-3QS1		-30.748	-14.644	-21.143				

6.3.3 L'algorithme hybride

Pour commencer ces tests, nous allons utiliser les mêmes valeurs qu'au début de la partie précédente : $\mathcal{P}_{vr} = 3$, $\mathcal{P}_K =$Q, $\mathcal{P}_S =$S1, $\mathcal{P}_{ns} = 5$ et $\mathcal{P}_{bbe} = 20$. Nous avons testé cet algorithme en même temps que les variantes de la partie précédente : c'est pour cela que nous n'utilisons pas les valeurs par défauts des paramètres. La figure 6.11 et le tableau 6.10 donnent un sommaire des résultats obtenus.

Dans la figure 6.11, la méthode hybride surpasse largement la version améliorée qui est elle-même supérieure à la variante originelle de PSD-MADS avec un grand écart. Pour $\tau = 10^{-3}$, l'approche hybride résout quasiment le double de problèmes que la version améliorée qui utilise k-*mean* uniquement. Dans le tableau 6.10, la méthode hybride donne les meilleures valeurs de Z_{worst} pour 9 problèmes parmi 11, ce qui montre que l'exploration de l'espace des solutions est meilleure en



(a) Profils de données ($\tau = 10^{-1}$). (b) Profils de données ($\tau = 10^{-2}$). (c) Profils de données ($\tau = 10^{-3}$).

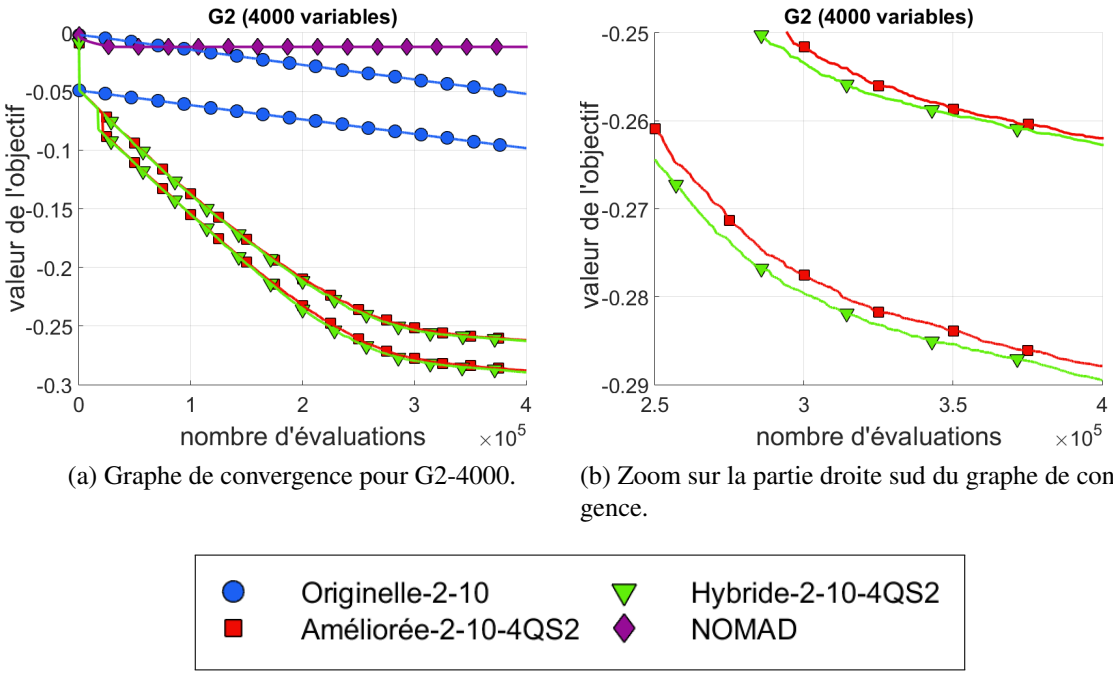
Figure 6.11 Comparaison de la version hybride aux versions originelle et améliorée de PSD-MADS.

Tableau 6.10 Sommaire des résultats de la comparaison entre les variantes originelle, améliorée et hybride.

Algo.	Prob.	Z_{best}	Z_{worst}	Z_{avg}	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-5-20	G2-500	-0.15432	-0.058469	-0.11804	CRESCENT-500	-40.52	-12.61	-19.453
Améliorée-5-20-3QS1		-0.26114	-0.22541	-0.23881		-55.54	**-17.01**	**-27.401**
Hybride-5-20-3QS1		**-0.26274**	**-0.22974**	**-0.2555**		**-103.98**	-16.19	-25.436
Originelle-5-20	G2-1000	-0.13701	-0.062465	-0.096506	CRESCENT-1000	-66.823	-17.37	-38.851
Améliorée-5-20-3QS1		-0.2581	-0.2331	-0.24885		**-171.971**	-44.524	**-88.469**
Hybride-5-20-3QS1		**-0.2606**	**-0.2483**	**-0.25318**		-120.333	**-50.616**	-79.514
Originelle-5-20	G2-2000	-0.10528	-0.066553	-0.086113	CRESCENT-2000	-120.354	-31.78	-64.303
Améliorée-5-20-3QS1		**-0.24533**	-0.22492	**-0.23985**		**-199.131**	**-84.175**	**-137.266**
Hybride-5-20-3QS1		-0.24344	**-0.23622**	-0.23975		-169.894	-82.831	-129.555
Originelle-5-20	DISK-500	-34.73	-29.1	-31.415	B250	9.754	120.996	31.894
Améliorée-5-20-3QS1		-28.482	-15.02	-22.027		**8.6219**	53.388	**19.429**
Hybride-5-20-3QS1		**-54.7**	**-44.16**	**-49.422**		8.9291	**53.242**	21.065
Originelle-5-20	DISK-1000	-36.13	-29.85	-33.797	B500	130.334	342.251	258.683
Améliorée-5-20-3QS1		-52.44	-16.298	-26.921		119.211	399.753	234.291
Hybride-5-20-3QS1		**-66.77**	**-53.99**	**-61.922**		**80.495**	**322.79**	**194.048**
Originelle-5-20	DISK-2000	-31.96	-30	-30.95				
Améliorée-5-20-3QS1		-30.783	-14.956	-21.954				
Hybride-5-20-3QS1		**-37.185**	**-30.398**	**-32.45**				

combinant *k-mean* avec la sélection aléatoire. Pour les problèmes DISK-500 et DISK-1000, la valeur Z_{worst} de la méthode hybride est meilleure que les valeurs Z_{best} des deux autres versions.

Pour le dernier test, nous allons utiliser les valeurs par défaut pour les cinq paramètres algorithmiques : $\mathcal{P}_{vr} = 4$, $\mathcal{P}_K$ =Q, $\mathcal{P}_S$ =S2, $\mathcal{P}_{ns} = 2$ et $\mathcal{P}_{bbe} = 10$. Nous comparons les versions originelle, améliorée et hybride en utilisant le problème G2 avec $4{,}000$ variables. Chaque algorithme est exécuté 30 fois avec le même point de départ et la figure 6.12 trace les courbes de convergence de la meilleure et de la pire instance uniquement ainsi qu'une exécution du problème par l'intermédiare de NOMAD. Le tableau 6.11 donne un sommaire des résultats.



(a) Graphe de convergence pour G2-4000.

(b) Zoom sur la partie droite sud du graphe de convergence.

Figure 6.12 Graphes de Convergence de la meilleure et pire exécution de chaque algorithme avec le problème G2-4000.

Tableau 6.11 Sommaire des résultats de la comparaison entre les 3 versions de PSD-MADS avec G2-4000.

Algorithme	Prob.	Z_{best}	Z_{worst}	Z_{avg}
Originelle-2-10	G2-4000	-0.098349	-0.052057	-0.079154
Améliorée-2-10-4QS2		-0.28788	-0.26198	-0.27218
Hybride-2-10-4QS2		**-0.28949**	**-0.26274**	**-0.27681**

Dans la figure 6.12(a), les méthodes améliorée et hybride surpassent la version originelle de PSD-MADS, mais la différence entre les deux nouvelles approches et à peine perceptible. Le tableau 6.11 et la figure 6.12(b) montrent que la version hybride est légèrement meilleure que la version améliorée.

6.4 Discussion

Ce chapitre vise à améliorer la méthode de sélection aléatoire de variables dans PSD-MADS. La méthode proposée se base sur une technique statistique pour construire la matrice de sensibilité qui quantifie l'influence des variables sur la fonction objectif et les contraintes du problème. L'algorithme *k-mean* est ensuite utilisé pour exploiter la matrice de sensibilité et en extraire des groupes

de variables nécessaires pour la construction des sous-problèmes. Une méthode hybride est aussi proposée qui alterne entre l'utilisation de k-*mean* et de la stratégie aléatoire pour faire la sélection des variables.

Les versions améliorée et hybride de PSD-MADS disposent de cinq paramètres qui contrôlent la construction de la matrice de sensibilité, les stratégies de groupement de réponses pour extraire plus efficacement des données de la matrice, le choix du paramètre k pour l'algorithme k-*mean*, la borne supérieure sur la taille des sous-problèmes et le nombre d'évaluations maximum alloués pour chaque sous-problème. Les tests numériques utilisent plusieurs valeurs des paramètres pour sélectionner des valeurs par défaut et les résultats montrent une nette amélioration par rapport à la version originelle de PSD-MADS, surtout avec le problème à $4{,}000$ variables où les stratégies améliorée et hybride surpassent largement la stratégie de sélection aléatoire.

Il faut noter que quelques tests préliminaires sur la matrice de sensibilité montrent qu'il est possible de détecter des corrélations entre les réponses d'un problème. Par exemple, une des matrices de sensibilité, générée par l'approche améliorée de PSD-MADS pour le problème G1 (13 variables et 9 contraintes) [126], détecte une corrélation entre les sixième et neuvième contraintes comme le montre la figure 6.13.

En revenant à la formulation analytique de ces deux contraintes :

$$\begin{array}{lrcrcrcl} \text{contrainte 6 :} & - & 8x_3 & + & x_{12} & & & \leq & 0 \\ \text{contrainte 9 :} & - & 8x_3 & + & x_{12} & - & x_9 & \leq & 0 \end{array} \tag{6.18}$$

nous pouvons constater que la corrélation est justifiée. Il est donc possible de concevoir une nouvelle stratégie qui se sert de la détection de corrélation pour le groupement des sorties.

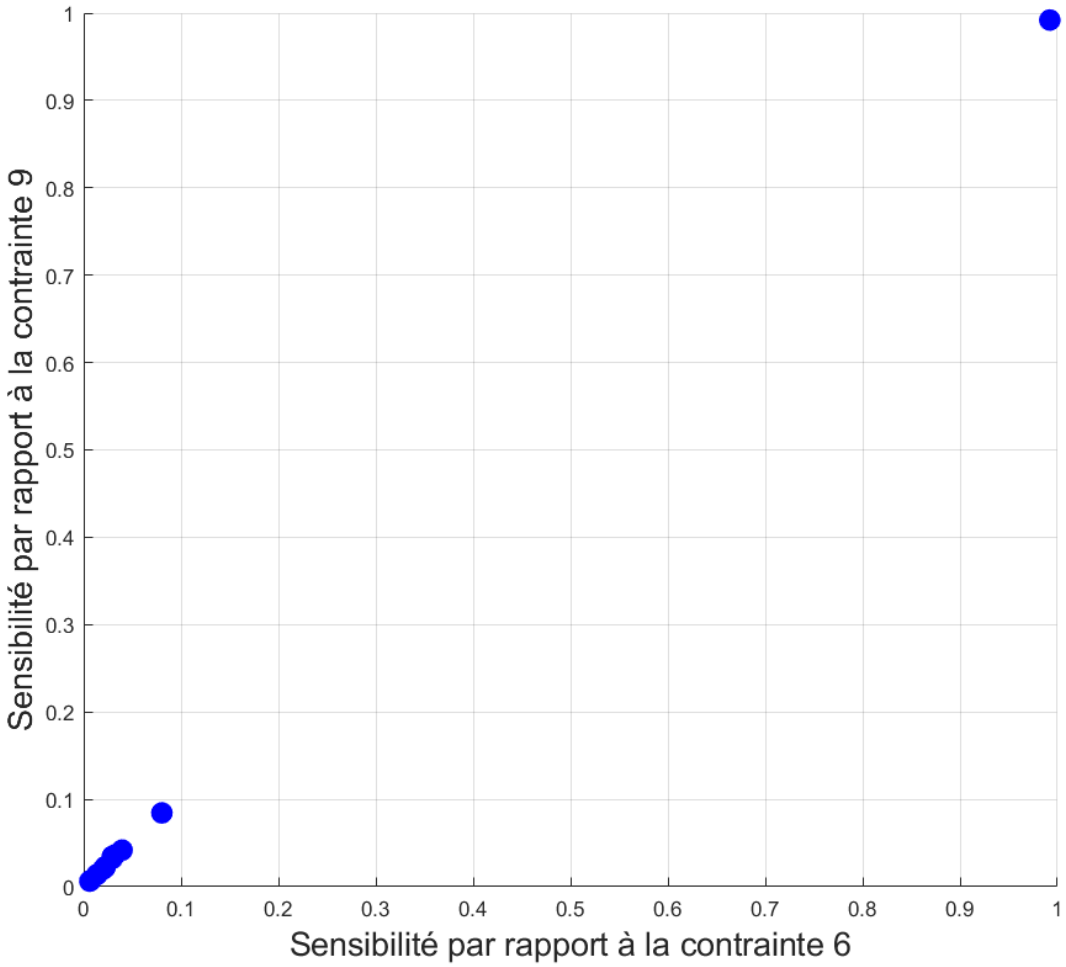


Figure 6.13 Détection d'une corrélation entre les sixième et neuvième contraintes du problème G1.

CHAPITRE 7 DISCUSSION GÉNÉRALE

Les travaux réalisés dans cette thèse ont mené à l'amélioration de deux extensions de MADS ainsi qu'à la création d'une nouvelle méthode d'optimisation non linéaire.

Tout d'abord, lors de l'utilisation des modèles quadratiques dans la phase de recherche de MADS, la résolution des sous-problèmes se faisait via une nouvelle instance de MADS. Nous utilisons deux méthodes d'optimisation non linéaire pour améliorer la résolution des sous-problèmes quadratiques : la fonction de pénalité en norme ℓ_1 et le Lagrangien augmenté. De plus, le Lagrangien augmenté en norme ℓ_1 est introduit comme une combinaison des points forts des deux méthodes précédentes.

L'objectif principal de cette partie est atteint. Il existe toujours une méthode qui donne de meilleurs résultats et qui roule plus vite que MADS. Les résultats numériques montrent aussi qu'il y a toujours une méthode, parmi les trois approches présentées dans ce chapitre, qui surpasse l'utilisation de MADS ou IPOPT pour l'optimisation des sous-problèmes. Ceci confirme que l'utilisation d'une méthode dédiée pour traiter les modèles quadratiques produit un NOMAD plus performant.

Une voie d'exploration possible est de considérer différents types de modèles pour construire les sous-problèmes dans la phase de recherche de MADS. Par exemple, il est possible d'utiliser des modèles quadratiques avec des contraintes linéaires ou bien tout simplement des modèles linéaires. Il est intéressant de comparer l'impact des différents modèles et de permettre à l'utilisateur de choisir le type de sous-problème à utiliser surtout lorsqu'il est confronté à une boite noire de dimension assez importante pour laquelle une interpolation quadratique devient très coûteuse. Il faut aussi s'intéresser à l'amélioration de la méthode du Lagrangien augmentée en norme ℓ_1 qui se révèle être l'approche la plus prometteuse pour la résolution des sous-problèmes quadratiques.

Il peut sembler étonnant de vouloir augmenter la méthode de pénalité exacte, mais les résultats numériques obtenus dans le chapitre 4 nous ont encouragés à explorer la nouvelle méthode du Lagrangien augmenté en norme ℓ_1 de manière détaillée en établissant une analyse de convergence. Nous prouvons la convergence globale de la méthode, en plus de nous assurer que le terme de pénalité est borné loin de zéro. De plus, nous adaptons l'approche de Coleman et Conn pour assurer une convergence superlinéaire en deux étapes de l'itération interne.

Notre implémentation du Lagrangien augmenté en norme ℓ_1 traite les contraintes d'inégalités directement. Donc, il faut vérifier comment le rajout de variables d'écart, comme c'est le cas pour le Lagrangien augmenté en norme ℓ_2, affecte l'implémentation de ℓ_1AUGLAG. En plus, l'approche de Coleman et Conn est basée sur une recherche linéaire. Donc, une autre voie à explorer est de rem-

placer cette approche dans l'itération interne par une méthode de région de confiance et comparer les deux différentes approches.

La seconde extension améliorée se rapporte à PSD-MADS. La sélection de variables pour construire des sous-problèmes était auparavant effectuée aléatoirement. Nous utilisons des outils statistiques pour quantifier l'influence des variables sur la fonction objectif et les contraintes d'un problème et nous la combinons avec l'algorithme de classification k-mean pour grouper les variables qui partagent à peu près la même influence sur les sorties. Cette technique peut soit être utilisée toute seule, soit combinée à la sélection aléatoire. Les nouvelles méthodes proposées surpassent la version de PSD-MADS qui utilise uniquement la sélection aléatoire des variables pour construire les sous-problèmes. Nous avons pu également augmenter la dimension des problèmes testés jusqu'à 4000 variables avec succès. Ces tests ont été effectués sur 500 coeurs du superordinateur CASIR d'Hydro-Québec.

Une suite naturelle est de rajouter une étape de recherche à la méthode MADS qui traite les sous-problèmes. Comme la taille des sous-problèmes est, en général, assez petite, il semble que l'utilisation de fonctions substituts dans l'étape de recherche devrait améliorer les résultats obtenus par les processus esclaves.

Dans la section 2.9, nous avons présenté la méthode PFA parce qu'elle présente une piste prometteuse qui doit être explorée. Le problème avec cette approche est qu'elle se base sur des multiplications et des décompositions de matrices denses et de grandes tailles, qui prennent un long temps d'exécution. Une solution serait de dédier un processus esclave pour exécuter l'algorithme PFA sur l'ensemble des points de la cache et construire une liste de groupes de variables.

CHAPITRE 8 CONCLUSION ET RECOMMANDATIONS

Notre intérêt pour la méthode du Lagrangien augmenté en norme ℓ_1 réside dans le fait qu'elle convient parfaitement pour les problèmes quadratiques avec contraintes quadratiques : cette approche transforme ces problèmes en une fonction quadratique par morceaux, ce qui, à notre avis, n'est pas considérablement plus difficile à résoudre que des problèmes quadratiques avec contraintes linéaires (qui représentent les sous-problèmes standards des méthodes d'optimisation non linéaire les plus rapides). Il faut noter, par contre, qu'à cause de toutes les projections requises, la méthode peut avoir un grand inconvénient au niveau du calcul pour les problèmes de grande dimension.

Les deux extensions de MADS décrites dans les deux autres projets pourront être intégrées à NOMAD. D'une part, l'utilisateur pourra choisir entre les différentes méthodes décrites dans le chapitre 4 pour résoudre les sous-problèmes quadratiques avec contraintes quadratiques lors de l'utilisation de MADS avec les modèles quadratiques. D'autre part, l'utilisateur pourra avoir le choix entre la stratégie aléatoire et les deux nouvelles stratégies introduites dans le chapitre 6 pour la construction des sous-problèmes dans PSD-MADS.

Pour intégrer toutes ces nouvelles méthodes dans NOMAD, il faut aussi déterminer les stratégies à utiliser par défaut dans MADS et PSD-MADS. Selon les résultats obtenus dans cette thèse, nous recommandons, tout d'abord, d'utiliser la méthode de Moré et Toraldo décrite dans le chapitre 4 comme l'algorithme par défaut pour traiter les sous-problèmes quadratiques sans contraintes dans l'étape de recherche globale de MADS dans NOMAD. Ensuite, nous conseillons d'utiliser la méthode du Lagrangien augmenté en norme ℓ_1 pour optimiser les sous-problèmes quadratiques avec contraintes quadratiques. Finalement, nous recommandons d'utiliser la stratégie hybride introduite dans le chapitre 6 comme méthode par défaut de décomposition parallèle de l'espace dans PSD-MADS.

Pour finir, les stratégies présentées dans le chapitre 6 sont conçues pour utiliser PSD-MADS dans de grands systèmes et peupler tous les processeurs disponibles. Cependant, pour des systèmes de plus petite taille, il faut définir de nouvelles stratégies pour exploiter le nombre limité de processeurs. Nous avons brièvement présenté deux de ces stratégies à la section 6.1.1. Nous avons aussi évoqué brièvement dans la conclusion du chapitre 6 qu'il est possible de détecter des corrélations entre les sorties du problème et d'établir des stratégies qui exploitent cette information.

RÉFÉRENCES

[1] K. Abhishek, S. Leyffer, et J.T. Linderoth. Modeling without categorical variables : a mixed-integer nonlinear program for the optimization of thermal insulation systems. *Optimization and Engineering*, 11(2) :185–212, 2010.

[2] C.M. Ablow et G. Brigham. An analog solution of programming problems. *Journal of the Operations Research Society of America*, 3(4) :388–394, 1955.

[3] M.A. Abramson, C. Audet, J.W. Chrissis, et J.G. Walston. Mesh Adaptive Direct Search Algorithms for Mixed Variable Optimization. *Optimization Letters*, 3(1) :35–47, 2009.

[4] M.A. Abramson, C. Audet, J.E. Dennis, Jr., et S. Le Digabel. OrthoMADS : A Deterministic MADS Instance with Orthogonal Directions. *SIAM Journal on Optimization*, 20(2) :948–966, 2009.

[5] L. Adjengue, C. Audet, et I. Ben Yahia. A variance-based method to rank input variables of the Mesh Adaptive Direct Search algorithm. *Optimization Letters*, 8(5) :1599–1610, 2014.

[6] J.S. Agte, J. Sobieszczanski-Sobieski, et R.R.J. Sandusky. Supersonic business jet design through bilevel integrated system synthesis. In *Proceedings of the World Aviation Conference*, volume SAE Paper No. 1999-01-5622, San Francisco, CA, 1999. MCB University Press, Bradford, UK.

[7] S. Alarie, C. Audet, V. Garnier, S. Le Digabel, et L.-A. Leclaire. Snow water equivalent estimation using blackbox optimization. *Pacific Journal of Optimization*, 9(1) :1–21, 2013.

[8] P. Alberto, F. Nogueira, H. Rocha, et L.N. Vicente. Pattern search methods for user-provided points : Application to molecular geometry problems. *SIAM Journal on Optimization*, 14(4) :1216–1236, 2004.

[9] N. Amaioua, C. Audet, A.R. Conn, et S. Le Digabel. Efficient solution of quadratically constrained quadratic subproblems within the mesh adaptive direct search algorithm. *European Journal of Operational Research*, 2017.

[10] S. Arreckx, A. Lambe, J.R.R.A. Martins, et D. Orban. A matrix-free augmented lagrangian algorithm with application to large-scale structural design optimization. *Optimization and Engineering*, 17(2) :359–384, 2016.

[11] C. Audet. Convergence Results for Generalized Pattern Search Algorithms are Tight. *Optimization and Engineering*, 5(2) :101–122, 2004.

[12] C. Audet. A survey on direct search methods for blackbox optimization and their applications. In P.M. Pardalos et T.M. Rassias, editors, *Mathematics without boundaries : Surveys in interdisciplinary research*, chapter 2, pages 31–56. Springer, 2014.

[13] C. Audet, V. Béchard, et S. Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2) :299–318, 2008.

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

[15] C. Audet et J.E. Dennis, Jr. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11(3) :573–594, 2001.

[16] C. Audet et J.E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3) :889–903, 2003.

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

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

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

[20] C. Audet, J.E. Dennis, Jr., et S. Le Digabel. Parallel space decomposition of the mesh adaptive direct search algorithm. *SIAM Journal on Optimization*, 19(3) :1150–1170, 2008.

[21] C. Audet, J.E. Dennis, Jr., et S. Le Digabel. Globalization strategies for Mesh Adaptive Direct Search. *Computational Optimization and Applications*, 46(2) :193–215, 2010.

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

[23] C. Audet, A. Ihaddadene, S. Le Digabel, et C Tribes. Robust optimization of noisy blackbox problems using the mesh adaptive direct search algorithm. *Optimization Letters*, Jan 2018.

[24] C. Audet, M. Kokkolaras, S. Le Digabel, et B. Talgorn. Order-based error for managing ensembles of surrogates in derivative-free optimization. *Journal of Global Optimization*, 70(3) :645–675, 2018.

[25] C. Audet et S. Le Digabel. The mesh adaptive direct search algorithm for periodic variables. *Pacific Journal of Optimization*, 8(1) :103–119, 2012.

[26] C. Audet, S. Le Digabel, et M. Peyrega. A derivative-free trust-region augmented Lagrangian algorithm. Technical Report G-2016-53, Les cahiers du GERAD, 2016.

[27] C. Audet, S. Le Digabel, et C. Tribes. Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization. *Optimization and Engineering*, 17(2) :333–358, 2016.

[28] C. Audet, G. Savard, et W. Zghal. A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research*, 204(3) :545–556, 2010.

[29] D.P. Bertsekas et J.N. Tsitsiklis. *Parallel and distributed computation : numerical methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[30] A.J. Booker, E.J. Cramer, P.D. Frank, J.M. Gablonsky, et J.E. Dennis, Jr. Movars : Multidisciplinary optimization via adaptive response surfaces. AIAA Paper 2007–1927, 2007.

[31] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.W. Moore, et D.B. Serafini. Managing surrogate objectives to optimize a helicopter rotor design – further experiments. AIAA Paper 1998–4717, Presented at the 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, 1998.

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

[33] E. Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6) :771–784, 2007.

[34] E. Borgonovo et E. Plischke. Sensitivity analysis : a review of recent advances. *European Journal of Operational Research*, 248(3) :869–887, 2016.

[35] J.M. Borwein et H.M. Strojwas. The hypertangent cone. *Nonlinear Analysis : Theory, Methods & Applications*, 13 :125–139, 1989.

[36] C.G. Broyden. The convergence of a class of double-rank minimization algorithms : 2. the new algorithm. *IMA Journal of Applied Mathematics*, 6(3) :222–231, 1970.

[37] C.-D. Caleanu, X. Mao, G. Pradel, S. Moga, et Y. Xue. Combined pattern search optimization of feature extraction and classification parameters in facial recognition. *Pattern Recognition Letters*, 32(9) :1250–1255, JUL 1 2011.

[38] L.A. Sarrazin-Mc Cann. Opportunisme et ordonnancement en optimisation sans dérivées. Mémoire de maîtrise, École Polytechnique de Montréal, 2018.

[39] C. Charalambous. A lower bound for the controlling parameters of the exact penalty functions. *Mathematical programming*, 15(1) :278–290, 1978.

[40] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[41] T.F. Coleman et A.R. Conn. Second-order conditions for an exact penalty function. *Mathematical Programming*, 19(1) :178–185, 1980.

[42] T.F. Coleman et A.R. Conn. Nonlinear programming via an exact penalty function : Asymptotic analysis. *Mathematical programming*, 24(1) :123–136, 1982.

[43] T.F. Coleman et A.R. Conn. Nonlinear programming via an exact penalty function : Global analysis. *Mathematical Programming*, 24(1) :137–161, 1982.

[44] A.R. Conn, N.I.M. Gould, et Ph.L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2) :545–572, 1991.

[45] A.R. Conn, N.I.M. Gould, et Ph.L. Toint. *LANCELOT : a Fortran package for large-scale nonlinear optimization (release A)*. Springer, New-York, 1992.

[46] A.R. Conn, N.I.M. Gould, et Ph.L. Toint. A note on exploiting structure when using slack variables. *Mathematical Programming*, 67(1) :89–97, 1994.

[47] A.R. Conn, N.I.M. Gould, et Ph.L. Toint. Numerical experiments with the LANCELOT package (release A) for large-scale nonlinear optimization. *Mathematical Programming*, 73(1) :73–110, 1996.

[48] A.R. Conn, N.I.M. Gould, et Ph.L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, 2000.

[49] A.R. Conn et 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.

[50] A.R. Conn, K. Scheinberg, et L.N. Vicente. Geometry of sample sets in derivative free optimization : Polynomial regression and underdetermined interpolation. *IMA Journal of Numerical Analysis*, 28(4) :721–749, 2008.

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

[52] A.R. Conn, L.N. Vicente, et C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. *SIAM Journal on Optimization*, 9(4) :924–947, 1999.

[53] R.I. Cukier, H.B. Levine, et K.E. Shuler. Nonlinear sensitivity analysis of multiparameter model systems. *Journal of computational physics*, 26(1) :1–42, 1978.

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

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

[56] J.E. Dennis, Jr. et V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4) :448–474, 1991.

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

[58] D. Echeverria Ciaurri, O.J. Isebor, et L.J. Durlofsky. Application of derivative-free methodologies to generally constrained oil production optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 2(2) :134–161, 2011.

[59] G. Fasano, G. Liuzzi, S. Lucidi, et F. Rinaldi. A linesearch-based derivative-free approach for nonsmooth constrained optimization. *SIAM Journal on Optimization*, 24(3) :959–992, 2014.

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

[61] M.C. Ferris et O.L. Mangasarian. Parallel variable distribution. *SIAM Journal on Optimization*, 4(4) :815–832, 1994.

[62] A.V. Fiacco et G.P. McCormick. *Nonlinear programming : sequential unconstrained minimization techniques*. Classics in Applied Mathematics 4. SIAM, 1990.

[63] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3) :317–322, 1970.

[64] R. Fletcher et S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, Series A, 91 :239–269, 2002.

[65] R. Fletcher et M.J.D. Powell. A rapidly convergent descent method for minimization. *The computer journal*, 6(2) :163–168, 1963.

[66] C. Fortin et H. Wolkowicz. The trust region subproblem and semidefinite programming. *Optimization Methods and Software*, 19(1) :41–67, 2004.

[67] K.R. Fowler, J.P. Reese, C.E. Kees, J.E. Dennis Jr., C.T. Kelley, C.T. Miller, C. Audet, A.J. Booker, G. Couture, R.W. Darwin, M.W. Farthing, D.E. Finkel, J.M. Gablonsky, G. Gray, et T.G. Kolda. Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5) :743–757, 2008.

[68] A. Frommer et R.A. Renaut. A unified approach to parallel space decomposition methods. *Journal of Computational and Applied Mathematics*, 110(1) :205–233, 1999.

[69] M. Fukushima. Parallel variable transformation in unconstrained optimization. *SIAM Journal on Optimization*, 8(3) :658–672, 1998.

[70] U.M. García-Palomares, I.J. García-Urrea, et P.S. Rodríguez-Hernández. On sequential and parallel non-monotone derivative-free algorithms for box constrained optimization. *Optimization Methods and Software*, 28(6) :1233–1261, 2013.

[71] U.M. García-Palomares et J.F. Rodríguez. New sequential and parallel derivative-free algorithms for unconstrained optimization. *SIAM Journal on Optimization*, 13(1) :79–96, 2002.

[72] H. Garg. Solving structural engineering design optimization problems using an artificial bee colony algorithm. *Journal of Industrial and Management Optimization*, 10(3) :777–794, 2014.

[73] A.E. Gheribi, C. Audet, S. Le Digabel, E. Bélisle, C.W. Bale, et A.D. Pelton. Calculating optimal conditions for alloy and process design using thermodynamic and properties databases, the FactSage software and the Mesh Adaptive Direct Search algorithm. *CALPHAD : Computer Coupling of Phase Diagrams and Thermochemistry*, 36 :135–143, 2012.

[74] A.E. Gheribi, C. Robelin, S. Le Digabel, C. Audet, et A.D. Pelton. Calculating all local minima on liquidus surfaces using the FactSage software and databases and the Mesh Adaptive Direct Search algorithm. *The Journal of Chemical Thermodynamics*, 43(9) :1323–1330, 2011.

[75] C.M. Giuliani et E. Camponogara. Derivative-free methods applied to daily production optimization of gas-lifted oil fields. *Computers & Chemical Engineering*, 75 :60–64, 2015.

[76] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109) :23–26, 1970.

[77] N.I.M. Gould, S. Lucidi, et Ph.L. Toint. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, 9(2) :504–525, 1999.

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

[79] N.I.M. Gould, D. Orban, et Ph.L. Toint. Galahad, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4) :353–372, 2003.

[80] N.I.M. Gould, D.P. Robinson, et H.S. Thorne. On solving trust-region and other regularised subproblems in optimization. *Mathematical Programming Computation*, 2(1) :21–57, 2010.

[81] R.B. Gramacy, G.A. Gray, S. Le Digabel, H.K.H. Lee, P. Ranjan, G. Wells, et S.M. Wild. Modeling an Augmented Lagrangian for Blackbox Constrained Optimization. *Technometrics*, 58(1) :1–11, 2016.

[82] G.A. Gray et T.G. Kolda. Algorithm 856 : APPSPACK 4.0 : Asynchronous parallel pattern search for derivative-free optimization. *ACM Transactions on Mathematical Software*, 32(3) :485–507, 2006.

[83] J.D. Griffin et T.G. Kolda. Asynchronous parallel hybrid optimization combining direct and gss. *Optimization Methods & Software*, 25(5) :797–817, 2010.

[84] J.D. Griffin, T.G. Kolda, et R.M. Lewis. Asynchronous parallel generating set search for linearly-constrained optimization. *SIAM Journal on Scientific Computing*, 30(4) :1892–1924, 2008.

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

[86] W.A. Gruver et E. Sachs. *Algorithmic Methods in Optimal Control*. Research Notes in Mathematics Series. Pitman Pub., 1981.

[87] H. Guo et R.A. Renaut. Parallel variable distribution for total least squares. *Numerical linear algebra with applications*, 12(9) :859–876, 2005.

[88] W.W. Hager. Minimizing a quadratic over a sphere. *SIAM Journal on Optimization*, 12(1) :188–208, 2001.

[89] C. Han, T. Feng, G. He, et T. Guo. Parallel variable distribution algorithm for constrained optimization with nonmonotone technique. *Journal of Applied Mathematics*, 2013, 2013.

[90] N. Hansen. The CMA Evolution Strategy : A Comparing Review. In J. Lozano, P. Larrañaga, I. Inza, et E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[91] A.-R. Hedar. Global Optimization Test Problems. `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm`. (last accessed on 2017-10-20).

[92] J.C. Helton. Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal. *Reliability Engineering & System Safety*, 42(2) :327–367, 1993.

[93] M.R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5) :303–320, 1969.

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

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

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

[97] R.A. Howard. Uncertainty about probability : A decision analysis perspective. *Risk Analysis*, 8(1) :91–98, 1988.

[98] A.K. Jain et R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[99] M. Jamil et X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2) :150–194, 2013.

[100] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[101] D.R. Jones, C.D. Perttunen, et B.E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1) :157–181, 1993.

[102] Chetehouna K., O. Sero-Guillaume, I. Sochet, et A. Degiovanni. On the experimental determination of flame front positions and of propagation parameters for a fire. *International Journal Of Thermal Sciences*, 47(9) :1148–1157, SEP 2008.

[103] A. Kannan et S.M. Wild. Benefits of Deeper Analysis in Simulation-based Groundwater Optimization Problems. In *Proceedings of the XIX International Conference on Computational Methods in Water Resources (CMWR 2012)*, June 2012.

[104] S. Kirkpatrick, C.D. Gelatt Jr., et M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598) :671–680, 1983.

[105] M. Kokkolaras, C. Audet, et J.E. Dennis, Jr. Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optimization and Engineering*, 2(1) :5–29, 2001.

[106] T.G. Kolda. Revisiting asynchronous parallel pattern search for nonlinear optimization. *SIAM Journal on Optimization*, 16(2) :563–586, 2005.

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

[108] J. Larson et S.M. Wild. A batch, derivative-free algorithm for finding multiple local minima. *Optimization and Engineering*, 17(1) :205–228, 2016.

[109] S. Le Digabel. Algorithm 909 : NOMAD : Nonlinear Optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4) :44 :1–44 :15, 2011.

[110] S. Le Digabel et S.M. Wild. A Taxonomy of Constraints in Simulation-Based Optimization. Technical Report G-2015-57, Les cahiers du GERAD, 2015.

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

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

[113] R.M. Lewis et V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4) :1075–1089, 2002.

[114] C.-S. Liu et C.-H. Tseng. Parallel synchronous and asynchronous space-decomposition algorithms for large-scale minimization problems. *Computational Optimization and Applications*, 17(1) :85–107, 2000.

[115] G. Liuzzi et K. Truemper. Parallelized hybrid optimization methods for nonsmooth problems using nomad and linesearch. *Computational and Applied Mathematics*, pages 1–36, 2017.

[116] Y. Lu, I. Cohen, X.S. Zhou, et Q. Tian. Feature selection using principal feature analysis. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 301–304. ACM, 2007.

[117] D.G. Luenberger. Control problems with kinks. *IEEE Transactions on Automatic Control*, 15(5) :570–575, 1970.

[118] L. Lukšan et J. Vlček. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical Report V-798, ICS AS CR, 2000.

[119] O.L. Mangasarian. Parallel gradient distribution in unconstrained optimization. *SIAM Journal on Control and Optimization*, 33(6) :1916–1925, 1995.

[120] A.L. Marsden, J.A. Feinstein, et C.A. Taylor. A computational framework for derivative-free optimization of cardiovascular geometries. *Computer Methods in Applied Mechanics and Engineering*, 197(21–24) :1890–1905, 2008.

[121] M.L. Martinez et J.C. Meza. On the use of direct search methods for the molecular conformation problem. *Journal of Computational Chemistry*, 15 :627–632.

[122] L.S. Matott, A.J. Rabideau, et J.R. Craig. Pump-and-treat optimization using analytic element method flow models. *Advances in Water Resources*, 29(5) :760–775, 2006.

[123] K. Matsui, W. Kumagai, et T. Kanamori. Parallel distributed block coordinate descent methods based on pairwise comparison oracle. *Journal of Global Optimization*, 69(1) :1–21, 2017.

[124] M.D. McKay, R.J. Beckman, et W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2) :239–245, 1979.

[125] Y. Mei, M.N. Omidvar, X. Li, et X. Yao. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software*, 42(2) :13, 2016.

[126] Z. Michalewicz et M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1) :1–32, 1996.

[127] M. Minville, D. Cartier, C. Guay, L.-A. Leclaire, C. Audet, S. Le Digabel, et J. Merleau. Improving process representation in conceptual hydrological model calibration using climate simulations. *Water Resources Research*, 50 :5044–5073, 2014.

[128] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, et M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, 151(2) :389–399, 2003.

[129] J.J. Moré et D.C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific Computing*, 4(3) :553–572, 1983.

[130] J.J. Moré et G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1) :93–113, 1991.

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

[132] M.D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2) :161–174, 1991.

[133] P. Mugunthan, C.A. Shoemaker, et R.G. Regis. Comparison of function approximation, heuristic and derivative-based methods for automatic calibration of computationally expensive groundwater bioremediation models. *Water Resources Research*, 41, 2005.

[134] I. Necoara et D. Clipici. Parallel random coordinate descent method for composite minimization : Convergence analysis and error bounds. *SIAM Journal on Optimization*, 26(1) :197–226, 2016.

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

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

[137] T.D. Plantenga. HOPSPACK 2.0 User Manual. Technical Report SAND2009-6265, Sandia National Laboratories, Livermore, CA, October 2009.

[138] T.K. Pong et H. Wolkowicz. The generalized trust region subproblem. *Computational Optimization and Applications*, 58(2) :273–322, 2014.

[139] M.J.D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, 1969.

[140] M.J.D. Powell. On trust region methods for unconstrained minimization without derivatives. *Mathematical Programming*, 97(3) :605–623, 2003.

[141] C.J. Price, I.D. Coope, et D. Byatt. A convergent variant of the Nelder Mead algorithm. *Journal of Optimization Theory and Applications*, 113(1) :5–19, 2002.

[142] N. V Queipo, R. T Haftka, W. Shyy, T. Goel, R. Vaidyanathan, et P.K. Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1) :1–28, 2005.

[143] R.G. Regis. Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2) :218–243, 2014.

[144] R.G. Regis. The calculus of simplex gradients. *Optimization Letters*, 9(5) :845–865, 2015.

[145] E. Renaud, C. Robelin, A.E. Gheribi, et P. Chartrand. Thermodynamic evaluation and optimization of the Li, Na, K, Mg, Ca, Sr // F, Cl reciprocal system. *Journal Of Chemical Thermodynamics*, 43(8) :1286–1298, 2011.

[146] F. Rendl et H. Wolkowicz. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77(1) :273–299, 1997.

[147] P. Richtárik et M. Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6) :1233–1243, 2016.

[148] C.A. Sagastizábal et M.V. Solodov. Parallel variable distribution for constrained optimization. *Computational Optimization and Applications*, 22(1) :111–131, 2002.

[149] A. Saltelli et J. Marivoet. Non-parametric statistics in sensitivity analysis for model output : a comparison of selected techniques. *Reliability Engineering & System Safety*, 28(2) :229–253, 1990.

[150] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[151] D.B. Serafini. *A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions*. Thèse de doctorat, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1998.

[152] S. Shan et G.G. Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 41(2) :219–241, 2010.

[153] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111) :647–656, 1970.

[154] J. Sobieszczanski-Sobieski, J.S. Agte, et R.R. Sandusky, Jr. Bi-Level Integrated System Synthesis (BLISS). Technical Report NASA/TM-1998-208715, NASA, Langley Research Center, 1998.

[155] I.M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Mathematical modelling and computational experiments*, 1(4) :407–414, 1993.

[156] I.M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1) :271–280, 2001.

[157] M.V. Solodov. New inexact parallel variable distribution algorithms. *Computational Optimization and Applications*, 7(2) :165–182, 1997.

[158] M.V. Solodov. On the convergence of constrained parallel variable distribution algorithms. *SIAM Journal on Optimization*, 8(1) :187–196, 1998.

[159] T. Pietrzykowski. An exact potential method for constrained maxima. *SIAM Journal on numerical analysis*, 6(2) :299–304, 1969.

[160] V. Torczon. *Multi-Directional Search : A Direct Search Algorithm for Parallel Machines*. Thèse de doctorat, Department of Mathematical Sciences, Rice University, Houston, Texas, 1989 ; available as Tech. Rep. 90-07, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005-1892.

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

[162] C. Tribes, J.-F. Dubé, et J.-Y. Trépanier. Decomposition of multidisciplinary optimization problems : formulations and application to a simplified wing design. *Engineering Optimization*, 37(8) :775–796, 2005.

[163] A. Wächter et L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1) :25–57, 2006.

[164] L. Wang, Y. Lei, Y. Zeng, L. Tong, et B. Yan. Principal feature analysis : A multivariate feature selection method for fmri data. *Computational and mathematical methods in medicine*, 2013, 2013.

[165] J. Xu, C. Audet, C.E. DiLiberti, W.W. Hauck, T.H. Montague, A.F. Parr, D. Potvin, et D.J. Schuirmann. Optimal adaptive sequential designs for crossover bioequivalence studies. *Pharmaceutical Statistics*, 15(1) :15–27, 2016.

[166] E. Yamakawa et M. Fukushima. Testing parallel variable transformation. *Computational Optimization and Applications*, 13(1–3) :253–274, 1999.

[167] W. Yang, J.A. Feinstein, et A.L. Marsden. Constrained optimization of an idealized y-shaped baffle for the fontan surgery at rest and exercise. *Computer Methods in Applied Mechanics and Engineering*, 199(33-36) :2135–2149, July 2010.

[168] W.I. Zangwill. Nonlinear Programming by sequential unconstrained maximization. Technical Report Working Paper 131, University of California, Berkeley, 1965.

[169] W.I. Zangwill. Non-linear programming via penalty functions. *Management science*, 13(5) :344–358, 1967.