UNIVERSITÉ DE MONTRÉAL

OPTIMISATION DE BOÎTES NOIRES MULTI-FIDÉLITÉ SOUS CONTRAINTES

XAVIER LEBEUF
DÉPARTEMENT DE MATHÉMATIQUES APPLIQUÉES ET GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE EN SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AVRIL 2023

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

OPTIMISATION DE BOÎTES NOIRES MULTI-FIDÉLITÉ SOUS CONTRAINTES

présenté par : LEBEUF Xavier
en vue de l'obtention du diplôme de : Maîtrise en sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. NOM Prénom, Doct., président
M. AUDET Charles, PhD., membre et directeur de recherche
M. LE DIGABEL Sébastien, PhD., membre et codirecteur de recherche
M. DIAGO-MARTÍNEZ Miguel, PhD., membre et codirecteur de recherche
M. NOM Prénom, PhD., membre

REMERCIEMENTS

Texte.

RÉSUMÉ

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

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

ABSTRACT

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

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

IREQ	Institut de recherche en électricité du Québec
GERAD	Groupe d'études et de recherche en analyse des décisions
PRIAD	Programme de robustesse, d'intégration et d'aide à la décision
DFO	Optimisation sans dérivée (*Derivative Free Optimization*)
BBO	Optimisation de boîtes noires (*Black Box Optimization*)
NOMAD	Optimisation non linéaire par recherche directe sur treillis adaptifs (*Nonlinear Optimisation by Mesh Adaptive Direct search*)
BP	Barrière progressive
BE	Barrière extrême
MADS	Recherche directe sur treillis adaptifs (*Mesh Adaptive Direct Search*)
MC	Monte-Carlo
PEB	Barrière progressive à extrême (*Progressive-to-Extreme Barrier*)

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

La recherche effectuée dans ce mémoire est motivée par un problème rencontré par l'Institut de recherche en électricité du Québec (IREQ), dans le cadre du projet programme de robustesse, d'intégration et d'aide à la décision (PRIAD) pour la gestion d'actifs d'Hydro-Québec TransÉnergie. L'Institut souhaite optimiser les périodicités de maintenance sur les équipements électriques du réseau électrique d'Hydro-Québec. L'un des objectifs de PRIAD est d'élaborer un simulateur du réseau de transport électrique d'Hydro-Québec qui prend en entrée les intervalles de temps où chaque type de maintenance est effectuée sur chaque sous-famille d'équipements de la partie du réseau simulée (ou du réseau en entier), pour associer un coût à cet ensemble d'intervalles, aussi appelé les périodicités de maintenances. À partir de ces dernières, des simulations d'indisponibilité des équipements ainsi que des simulations électriques en considérant les indisponibilités sont effectuées. Le coût est ensuite calculé en considérant, entre autres, le coût de la maintenance, le nombre de défaillances concourantes, la quantité d'énergie non livrée et la gravité des pannes. Le simulateur est détaillé et illustré à la section 2.5. L'IREQ souhaite trouver les périodicités qui minimisent le coût. L'intuition derrière cet objectif est qu'il est possible d'imaginer qu'effectuer toutes les maintenances sur le réseau au complet quotidiennement coûterait beaucoup trop cher, et inversement, trop peu de maintenance causerait beaucoup de défaillances sur le réseau. L'objectif est d'optimiser les périodicités pour trouver celles qui font le meilleur compromis, c'est-à-dire celles qui minimisent le coût.

Il est impossible d'extraire un ensemble d'équations formant un modèle mathématique qui définit le comportement du simulateur. Il n'est donc pas possible d'utiliser les méthodes d'optimisation qui font usage d'expressions analytiques ou qui supposent l'existence des dérivées. Conséquemment, les méthodes d'optimisation sans dérivées : *Derivative Free Optimization* (DFO) et d'optimisation de boîtes noires : *Black Box Optimization* (BBO) doivent être mises à l'usage. En effet, le simulateur peut être vu comme un procédé qui prend des périodicités en entrée et qui renvoie un coût et des conditions, considérés respectivement comme l'objectif à optimiser et des contraintes mathématiques. Par exemple, une contrainte pourrait être que lors des simulations électriques, le nombre d'évènements où plusieurs équipements sont simultanément indisponibles doit être inférieur ou égal à une certaine quantité. Formulé ainsi, le procédé peut être vu comme une boîte noire, et il s'agit d'un cas typique d'application de méthodes de BBO. Appliquer de telles méthodes avec un logiciel existant est généralement relativement simple; le simulateur de l'IREQ pose toutefois deux problèmes particuliers. Le premier est qu'il contient une simulation Monte-Carlo (MC), signifiant que la simulation est

stochastique. À chaque lancement de la simulation, la méthode doit déterminer un niveau de précision avec lequel effectuer la simulation. Ce problème peut être contourné en demandant une haute qualité aux résultats en effectuant beaucoup de tirages MC, mais cette pratique cause le second problème. Les méthodes de BBO sont des méthodes itératives qui lancent la simulation à répétition. Si chaque simulation est longue lorsqu'on y effectue beaucoup de tirages, le temps total d'optimisation devient très grand. Dans le cas de PRIAD, le temps d'exécution d'une simulation d'un réseau de transport de 2000 équipements avec 5×10^5 tirages MC est de 145 jours. De ce fait, le temps requis par une méthode d'optimisation qui appelle une telle simulation 2000 fois est d'environ 800 ans.

1.1 Objectifs de recherche et plan du mémoire

Au moment de l'écriture de ce mémoire, le simulateur de PRIAD est toujours en développement. L'objectif premier de ce projet de maîtrise est de développer une méthode générale de réduction du temps d'optimisation, qui sera appliquée dans des travaux futurs à la problématique d'Hydro-Québec. Conséquemment, la méthode est développée pour exploiter des propriétés que la boîte noire de PRIAD comporte, et elle est applicable à toute boîte noire qui comporte également ces propriétés. Les propriétés en question sont la multi-fidélité ainsi que l'existence de contraintes difficiles à satisfaire.

Comme la boîte noire de l'IREQ comprend plusieurs contraintes rarement satisfaites, il arrive que durant une optimisation, un grand temps soit investi inutilement dans des évaluations qui donnent de mauvaises solutions. La méthode présentée dans ce mémoire vise à réduire ce temps, en utilisant des évaluations à basse fidélité, donc plus rapides, pour déterminer si un point vaut les ressources d'une évaluation plus longue. Il s'ensuit que plus les contraintes sont difficiles à satisfaire, plus il y a de temps à sauver avec la méthode proposée. Inversement, une boîte noire sans contraintes ou sans aspect multi-fidélité ne peut bénéficier de cette méthode.

Plusieurs autres travaux de recherche visent à effectuer des optimisations à précision variable. En revanche, ces travaux s'intéressent le plus souvent à l'impact de la précision sur la valeur de l'objectif. Ces travaux visent à guider des algorithmes d'optimisation pour trouver à quelle fidélité effectuer chaque évaluation pour trouver de meilleures solutions. Le projet de maîtrise qui fait l'objet de ce mémoire s'intéresse plutôt à un autre aspect qui manque dans la littérature : l'impact de la précision variable sur les valeurs des contraintes, et l'utilisation de l'information que ces dernières peuvent donner à basse fidélité pour réduire le temps de calcul. En effet, la méthode présentée ici considère un ensemble discret de fidélités auxquelles sont assignées les contraintes. Les contraintes sont hiérarchisées selon les valeurs des fidélités assignées. Des travaux futurs pourront étudier la possibilité d'agencer la méthode de hiérar-

chie de groupes de contraintes aux méthodes d'optimisation multi-fidélité qui s'intéressent seulement à l'objectif pour proposer de nouvelles méthodes d'optimisation potentiellement très efficaces.

Un second objectif est de tester la méthode développée avec quelques implémentations différentes. La méthode d'optimisation avec contraintes hiérarchisée comprend un algorithme développé pour agir en tant que passerelle entre un algorithme d'optimisation et la boîte noire à optimiser. De ce fait, la méthode doit être couplée avec un autre algorithme d'optimisation de boîtes noires existant. De cette façon, les propriétés de convergence ainsi que les performances d'algorithmes qui ont déjà fait leurs preuves peuvent être conservées, et la passerelle peut effectuer des interruptions en cours d'évaluation indépendamment de cet algorithme. Dans ce projet, la méthode est testée avec l'algorithme de recherche directe sur treillis adaptifs : *Mesh Adaptive Direct Search* (MADS), et avec différentes méthodes de gestion des contraintes. Le logiciel optimisation non linéaire par recherche directe sur treillis adaptifs : *Nonlinear Optimisation by Mesh Adaptive Direct search* (NOMAD) est utilisé pour les tests.

*** plan du mémoire *** Je vais attendre de voir à quoi va ressembler la table des matières finale pour ce paragraphe.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce second chapitre vise à établir les notions pré-requises à la compréhension de ce projet, ainsi qu'à décrire les avancements récents dans divers champs de l'optimisation de boîtes noires connexes à ce projet. Les notions de base sur l'optimisation de boîtes noires, la gestion des contraintes, la multi-fidélité, le projet d'Hydro-Québec PRIAD, l'étude des boîtes noires stochastiques, les méthodes de réduction du temps d'optimisation ainsi que la théorie des graphes y sont couverts. Il est à noter que dans les chapitres subséquents, PRIAD ne sera que très peu mentionné pour les raisons mentionnées dans l'introduction. Il est tout de même important de couvrir le sujet dans la revue car ce sont les technicalités de ce projet qui ont forgé la méthode présentée dans ce mémoire. Les propriétés pouvant être exploitées par la méthode sont données par ce projet. La méthode est ensuite dite générale car elle s'applique à toute boîte noire qui comporte ces propriétés.

2.1 Optimisation de boîtes noires

Cette section aborde le sujet de la BBO telle que décrite par Audet et Hare (2017). L'optimisation de boîtes noires est une branche de la recherche opérationnelle en mathématiques appliquées qui s'intéresse aux problèmes où certains éléments n'ont pas de formulation analytique. Typiquement, aucune hypothèse n'est posée sur la continuité, la différentiabilité, et le caractère lisse de la fonction objectif et des contraintes. Une boîte noire prend un ou plusieurs paramètres en entrée, et applique un procédé qui renvoie la valeur de la fonction objectif ainsi que les valeurs des contraintes, si applicable. Par exemple, un code informatique ou une expérience en laboratoire peut correspondre à cette définition. En particulier, ce document s'intéresse au cas mono-objectif contraint. Les problèmes multi-objectifs ne sont pas considérés.

Il est à noter que tout problème d'optimisation formulé en modèle analytique correspond également à cette définition, mais les méthodes qui exploitent l'information sur les dérivées sont généralement beaucoup plus efficaces que les méthodes de BBO. Une évaluation d'une boîte noire est définie comme l'action requise pour obtenir les valeurs des sorties étant donnés certains paramètres. Le tableau 2.1 liste les définitions des expressions mathématiques importantes.

Expression	Définition
$n \in \mathbb{N}^*$	Dimension du problème : nombre de paramètres d'entrée
$m \in \mathbb{N}$	Nombre de contraintes relaxables du problème
$X \subseteq \mathbb{R}^n$	Espace des paramètres sur lequel f est définie
$x \in X$	Point, vecteur contenant les valeurs des paramètres
$f : X \to \mathbb{R} \cup \{\infty\}$	Fonction qui renvoie la valeur de l'objectif au point x
$c : X \to \mathbb{R}^m \cup \{\infty\}^m$	Fonction qui renvoie les valeurs des contraintes relaxables au point x
$c_j(x) \leq 0$	j-ième contrainte relaxable, où $j \in J := \{1, 2, \ldots, m\}$

Tableau 2.1 Définitions liées à l'optimisation de boîtes noires.

Pour le reste du mémoire, par souci d'alléger l'écriture, l'abus de langage c_j est utilisé pour référer à une contrainte $c_j(x) \leq 0$. En résumé, une boîte noire prend en entrée x et renvoie $f(x)$ et $c(x)$. L'équation (2.1) donne la formulation générale d'un problème d'optimisation.

$$\min_{x \in X} f \quad \text{s.c.} \quad x \in \Omega = \{x \in X : c_j(x) \leq 0, j \in J\}. \tag{2.1}$$

Un point x est dit réalisable si toutes les contraintes y sont satisfaites, et Ω dénote l'ensemble des solutions réalisables. L'ensemble X est défini par les contraintes non relaxables. Une contrainte est dite non relaxable si elle doit être satisfaite par toute solution. Typiquement, X est borné par les limites inférieure $l \in \mathbb{R}^n$ et supérieure $u \in \mathbb{R}^n$ de chaque paramètre. Par exemple, un paramètre représentant une probabilité doit appartenir à l'ensemble $[0, 1]$. Ces contraintes non relaxables ne renvoient pas nécessairement un nombre, elles pourraient renvoyer un message d'erreur par exemple. Lorsqu'elles ne sont pas respectées, toutes les sorties sont potentiellement erronées. Inversement, les contraintes relaxables n'ont pas d'impact sur la validité des autres sorties, et elles renvoient toujours une mesure de la réalisabilité d'un point. Posons à titre d'exemple une contrainte qui signifie qu'un budget ne doit pas dépasser b\$. Si un point x est tel que le budget excède b\$, la contrainte indiquera de combien le budget a été dépassé, et toutes les autres sorties sont tout de même valides. Il est posé par convention que l'objectif est minimisé. Sans perte de généralité, tout problème de maximisation peut se réécrire en problème de minimisation en changeant le signe de l'objectif.

Comme l'information sur les relations entre les entrées et les sorties d'une boîte noire est supposée inaccessible, les algorithmes de BBO doivent obtenir l'information en exécutant

la boîte noire. Plusieurs évaluations sont effectuées consécutivement, et chaque point x^k à évaluer est déterminé par l'algorithme selon les sorties observées aux évaluations précédentes, à l'exception du premier point. L'indice k indique qu'il s'agit du k-ième point à évaluer. Le point de départ est alors appelé x^0. L'algorithme garde en mémoire un point champion x^*. À chaque itération, lorsqu'un point pour lequel la valeur de f est la plus petite tout en respectant les contraintes est trouvé, x^* est mis à jour. À l'atteinte d'un critère d'arrêt, l'algorithme renvoie x^*. La figure 2.1 illustre comment un solveur qui applique un algorithme de BBO effectue des appels de la boîte noire.

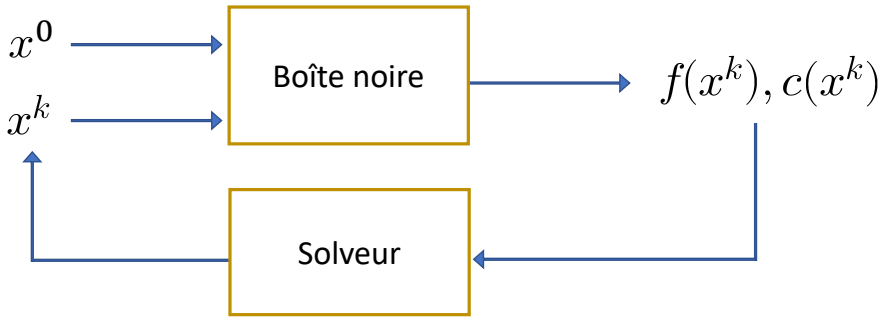


FIGURE 2.1 Généralisation des méthodes d'optimisation de boîte noire.

Les algorithmes de BBO sont divisés en trois grandes catégories. La première consiste en les méthodes qui génèrent des substituts de la boîte noire. Un substitut est un système peu coûteux à évaluer qui approxime le comportement de la boîte noire. Le cadre d'optimisation de substitus entiers-mixtes : *Mixte-Integer Surrogate Optimisation framework* (MISO) présenté par Müller (2016) est un exemple d'implémentation d'une méthode qui correspond à cette catégorie. Il s'agit d'une implémentation MATLAB spécialisée pour les boîtes noires coûteuses en entier-mixte. La seconde comprend les méthodes de recherche directe, qui utilisent directement l'information des évaluations. Quelques-unes de ces méthodes sont décrites ci-dessous. Plusieurs logiciels d'optimisation utilisent des algorithmes provenant de ces deux catégories conjointement. Finalement, il y a les méthodes heuristiques, qui ne sont pas discutées dans ce mémoire.

Il existe plusieurs algorithmes d'optimisation qui sont reconnus pour leur efficacité. Entre autres, l'algorithme MADS proposé par Audet et Dennis, Jr. (2006) se distingue par ses propriétés de convergence globale vers des minimums locaux. Il s'agit d'une amélioration directe de l'algorithme Recherche par motifs généralisée : *Generalized Pattern Search* proposé par Torczon (1997) où à partir du point itéré courant, chaque nouveau point est généré en

effectuant un pas dans la direction d'une coordonnée de l'espace des paramètres. À chaque itération, $2n$ points à évaluer sont alors générés. Cette méthode est problématique lorsqu'un pas dans une combinaison de directions de différentes coordonnées doit être effectué pour atteindre un minimum. Avec MADS, les directions sont générées aléatoirement. Pour conserver les propriétés de convergence, les directions forment une base positive et donnent des points positionnés sur un treillis adaptatif. L'algorithme de Nelder et Mead (1965) place les points à évaluer sur un simplexe (un polytope à $n+1$ arètes). Dépendamment des évaluations, certains points du simplexe sont déplacés en effectuant une réflexion, une expansion, une contraction ou une réduction. Les résultats de cet algorithme sont souvent très dépendants du simplexe de départ. Huyer et Neumaier (1999) ont proposé l'algorithme Recherche par coordonnées multi-niveaux : *Multilevel Coordinate Search* qui consiste à diviser itérativement l'espace des paramètres en hyperrectangles. À chaque itération, un point est évalué et un hyperrectangle est divisé le long d'un axe auquel le point appartient et déterminé par le résultat de l'évaluation. La quantité d'algorithmes de DFO et BBO est vaste, et la quantité d'implémentations logicielles différentes de ces algorithmes est tout aussi vaste. De ce fait, tel que démontré par Rios et Sahinidis (2013), il n'est pas évident de déterminer quelles implémentations de quels algorithmes sont les meilleures. Une total de 22 logiciels différents ont été testés sur 502 problèmes pour découvrir que tout solveur a trouvé la meilleure solution pour au moins quelques problèmes, et qu'aucun solveur domine tous les autres. Plus récemment, une analyse similaire a été effectuée par Ploskas et Sahinidis (2022) en s'intéressant aux problèmes entier-mixtes en particulier. Les conclusions ne sont pas les mêmes ; l'étude a trouvé que NOMAD et MISO se démarquent de façon évidente.

Les quelques algorithmes présentés dans cette section sont plutôt vieux, mais des travaux de recherche sur ceux-ci continuent à générer des réflexions à leur sujet et à les améliorer ainsi que leurs implémentations. Conséquemment, ils sont encore fort pertinents aujourd'hui. Par exemple, Audet et al. (2022a) adaptent MADS pour tenir compte de problèmes où les meilleures solutions sont près de régions discontinues. Aussi, Ozaki et al. (2019) suggèrent un algorithme de parallélisation de la méthode de Nelder-Mead pour l'accélérer, tout en ajoutant des évaluations spéculatives basées sur un modèle, également effectuées en parallèle. Un modèle est un substitut qui tente d'être le plus fidèle possible à la boîte noire. Toujours au sujet de l'algorithme de Nealer-Mead, Galántai (2022) propose une preuve de convergence qui s'applique aux espaces dont la dimension n'excède pas huit. Alarie et al. (2021a) décrivent un large éventail d'applications en BBO qui ont eu lieu dans les 20 dernières années, principalement dans les domaines de l'énergie, de la science des matériaux et du génie informatique.

La méthode présentée dans de ce mémoire est testée avec le logiciel NOMAD présenté par

Audet et al. (2022b). Il s'agit d'une implémentation de MADS qui vise particulièrement à optimiser des boîtes noires données par des programmes très couteux en temps où le budget d'évaluations accordé à une optimisation est limité. Cette version récente est l'évolution du logiciel décrit par Le Digabel (2011).

2.2 Gestion des contraintes

Cette section décrit d'abord deux méthodes de gestion des points non réalisables au cours de l'application d'un algorithme d'optimisation. Elles permettent aussi de traiter les cas où le tout premier point, généralement donné par l'utilisateur, n'est pas réalisable. D'abord, définissons la fonction $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ appelée la fonction de violation des contraintes, inspirée des méthodes par filtres de Gould et Toint (2010).

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

Avec cette définition, h est une fonction non négative qui renvoie 0 si $x \in \Omega$, et qui caractérise par quelle mesure x ne respecte pas les contraintes relaxables si $x \notin \Omega$ et $x \in X$. Les valeurs des contraintes sont généralement mises à l'échelle pour que leur grandeur soit comparable. Si la fonction avait été définie sans le carré, mais plutôt avec une norme l_1, il y aurait introduction de non différentiabilité. Le carré permet à la fonction d'être non négative tout en conservant certaines propriétés de différentiabilité. De plus, la fonction avec le carré se comporte mieux dans un contexte algorithmique où l'on désire que la valeur de cette fonction atteigne zéro.

La première méthode se nomme la barrière extrême (BE). Cette méthode se définit par deux phases. La première vise à trouver un premier point réalisable en minimisant la fonction de violation des contraintes et en ignorant totalement la valeur de l'objectif. Cette première phase est utile uniquement lorsque le premier point n'est pas réalisable. La seconde phase optimise le problème en ignorant les points non réalisables. La méthode est détaillée à l'Algorithme 2.1.

Cette façon de traiter les contraintes relaxables est très simple, et elle n'utilise pas l'information sur les points non-réalisables à partir de la seconde phase. La seconde méthode, proposée par Audet et Dennis, Jr. (2009), se nomme la barrière progressive, et elle tente

Algorithme 2.1 : Barrière extrême à deux phases, tiré de Audet et Hare (2017)

Étant donné $f_\Omega : X \to \mathbb{R} \cup \{\infty\}, c : X \to \mathbb{R}^m \cup \{\infty\}^m$ et un point de départ $x^0 \in X$

1. Phase de réalisabilité

   Lancer un algorithme d'optimisation à partir de x^0 pour résoudre $\min_{x \in X} h(x)$.

   Terminer l'optimisation dès qu'un point réalisable $\bar{x} \in \Omega$ est trouvé, puis aller à 2.

   Si un autre critère d'arrêt est satisfait avant de trouver ce $\bar{x}$,

   terminer en concluant qu'aucun point appartenant à Ω n'a été trouvé.

2. Phase d'optimisation

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

   Lancer un algorithme d'optimisation à partir de $\bar{x}$ pour résoudre $\min_{x \in X} f_\Omega$

d'utiliser cette information pour potentiellement trouver de meilleurs points réalisables tout au long de l'optimisation. La barrière progressive (BP) introduit un seuil $h^k_{\max} \in \mathbb{R}_+$, mis à jour à chaque itération, où les itérations sont dénotées par k. Chaque x où $h(x) > h^k_{\max}$ est ignoré en attribuant à $f(x)$ une valeur de ∞. Le seuil initial est donné par $h^0_{\max} = \infty$, et le seuil est monotone décroissant avec les itérations.

Au lieu de garder en mémoire un seul point itéré avec la meilleure valeur de f, la méthode propose de garder deux itérés : un point réalisable et un point non réalisable, nommés respectivement x^{fea} et x^{inf}. Les valeurs de l'objectif $f(x^{\text{fea}})$ et $f(x^{\text{inf}})$ sont initialisées à ∞, et les points sont mis à jour par l'Algorithme 2.2.

Algorithme 2.2 : Mise à jour des points x^{fea} et x^{inf} de la barrière progressive

Étant donné x^{fea}, x^{inf} et un nouveau point x récemment évalué

Si $h(x) = 0$ et $f(x) < f(x^{\text{fea}})$

   $x^{\text{fea}} \leftarrow x$

Si $f(x) \leq f(x^{\text{inf}})$ et $h(x) \leq h(x^{\text{inf}})$ et $\Big(f(x) < f(x^{\text{inf}})$ ou $(x) < h(x^{\text{inf}})\Big)$

   $x^{\text{inf}} \leftarrow x$

L'algorithme d'optimisation couplé avec la BP explore alors autour de x^{fea} et de x^{inf}. L'intérêt est que x^{inf} est souvent un point avec une bien meilleure valeur de f que x^{fea}, et en faisant progresser la valeur de $h^k_{\max}$ vers 0, il est possible de trouver un point réalisable près de x^{inf} très intéressant. La description de la barrière est intentionnellement vague car son application dépend de l'algorithme d'optimisation avec lequel elle est couplée.

Il est à noter que pour un même problème d'optimisation, il est possible d'appliquer différentes barrières aux différentes contraintes. Audet et al. (2010) présentent une troisième méthode, nommée la barrière progressive à extrême : *Progressive-to-Extreme Barrier* (PEB). Cette approche est utile pour les optimisations avec des points de départ non réalisables où l'on souhaite appliquer la BP seulement à la phase de réalisabilité de la BE. L'algorithme débute en appliquant la BP à toutes les contraintes. À chaque fois qu'un point évalué est tel qu'une contrainte qui n'était pas satisfaite précédemment le devient, cette contrainte est maintenant traitée avec la BE. Une fois arrivé à la phase d'optimalité, toutes les contraintes sont traitées par la BE.

Dans la littérature récente, Papalexopoulos et al. (2022) proposent d'utiliser un réseau de neurones et un programme linéaire entier-mixte pour former un modèle qui peut traiter plus aisément les contraintes discrètes. Dans le cas de boîtes noires stochastiques, ces modèles sont souvent utilisés. Toutefois, Dzahini et al. (2022) suggèrent une modification à MADS où, au lieu d'utiliser des modèles, des estimations de fonctions et des bornes probabilistes avec des conditions suffisantes de descente permettent d'optimiser les cas contraints et stochastiques. Aussi, Audet et al. (2022c) analysent l'efficacité de la BP avec NOMAD en utilisant la plateforme COCO décrite par Hansen et al. (2021). Pour une meilleure utilisation de la BP, Audet et al. (2018a) proposent un algorithme de régions de confiance (construction de modèles locaux) où deux régions sont construites : une première autour de x^{fea} et une seconde autour de x^{inf}. D'autre part, Bajaj et al. (2018) combinent un algorithme de régions de confiance avec une méthode comprenant une phase de réalisabilité et une phase d'optimisation très semblable à la BE.

D'un point de vue général, les contraintes peuvent être classifiées en neuf catégories selon Le Digabel et Wild (2015). La figure 2.2 illustre en un arbre une série de questions à se poser pour classifier une contrainte en partant de la racine. Chaque feuille de l'arbre correspond à une classe. D'abord, une contrainte peut être connue ou cachée. Le terme « contrainte cachée » a été introduit par Chen et Kelley (2016). Une contrainte est connue si elle est explicitement donnée dans la formulation du problème. Une contrainte cachée peut s'agir d'un bogue informatique, ou elle peut simplement être $x > 0$ dans le problème $\min\{\log(x) : x \in \mathbb{R}\}$ si la contrainte n'est pas indiquée au solveur. Une contrainte cachée est nécessairement une

contrainte de simulation non relaxable et non quantifiable. Une contrainte est dite à priori si elle ne requiert pas le lancement d'une simulation pour vérifier sa réalisabilité. Cette définition inclut tous les problèmes formulés analytiquement, et les contraintes par simulation sont propres à l'optimisation de boîtes noires. La relaxabilité a été discutée à la section 2.1. Finalement, une contrainte est quantifiable si elle renvoie une mesure de la réalisabilité de la contrainte. Une contrainte non quantifiable renvoie une quantité binaire, qui indique si elle est satisfaite ou non, sans savoir à quel degré.

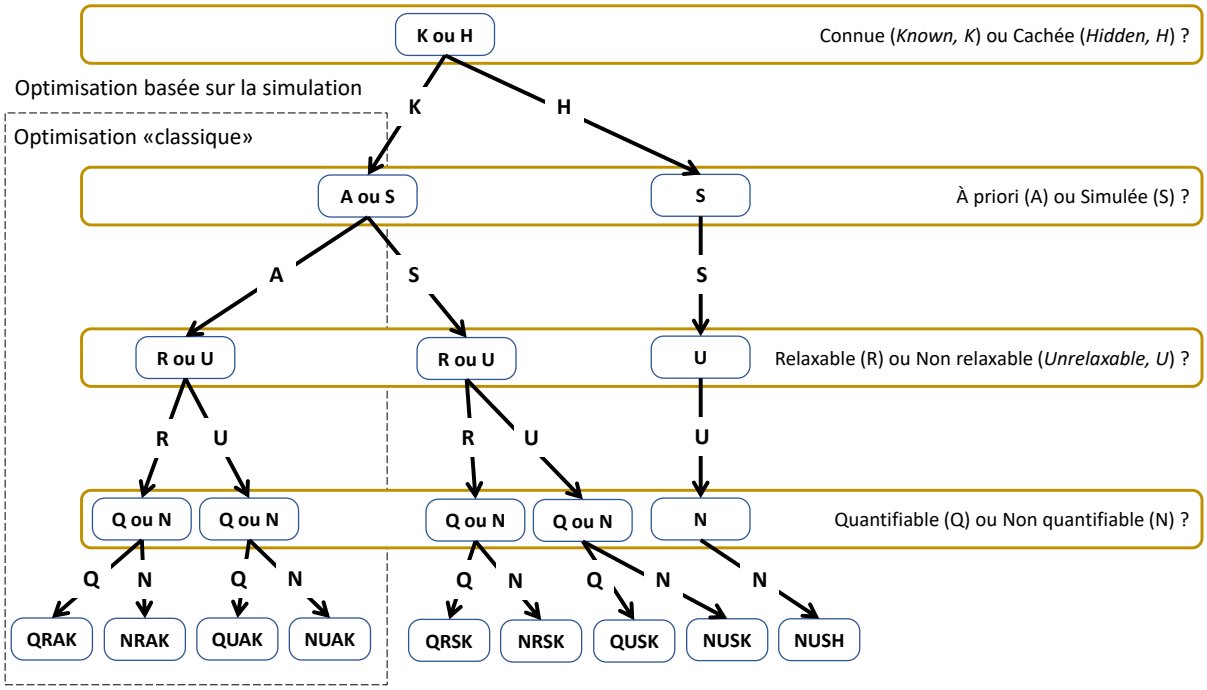


FIGURE 2.2 Représentation en arbre de la taxonomie des contraintes QRAK traduite de Le Digabel et Wild (2015).

2.3 Multi-fidélité

Cette section décrit le concept de multi-fidélité ainsi que ses implications dans un contexte d'optimisation de boîtes noires. Une fidélité est une mesure de la qualité des sorties d'un procédé et du coût en temps pour atteindre cette qualité. Une basse fidélité offre un résultat peu coûteux mais de qualité dégradée, alors qu'une haute fidélité correspond à des sorties très fiables mais coûteuses à obtenir. Il est à noter que la fidélité peut être définie de manière indépendante du coût, ce n'est cependant pas le cas pour ce projet de maîtrise. Un procédé est dit multi-fidélité s'il peut être effectué à différents niveaux de fidélité. Une grande branche de la BBO s'intéresse à la bi-fidélité, un cas particulier de la multi-fidélité où la vérité et une

approximation de la vérité sont disponibles. Souvent, le procédé lui-même donne la vérité (haute-fidélité) et il existe un substitut qui donne une approximation (basse-fidélité). Il est aussi possible que la haute et la basse fidélité proviennent toutes deux du même procédé, en variant certains paramètres. En s'appliquant à la multi-fidélité, la méthode présentée dans ce mémoire s'applique aussi à la bi-fidélité, mais les méthodes spécialisées en bi-fidélité comme celle de Balabanov et Venter (2004) risquent d'être plus efficaces lorsque seulement deux fidélités sont disponibles.

Pour citer un exemple de procédé multi-fidélité intéressant pour ce projet, une simulation MC peut être effectuée à différentes fidélités en variant le nombre de tirages effectués. En augmentant le nombre de tirages, la précision de la simulation et le temps de simulation augmentent et la fidélité est dite plus grande. Inversement, la fidélité diminue en diminuant le nombre de tirages. Le tableau 2.2 liste les définitions importantes liées à la multi-fidélité, basées sur celles présentées par Sen et al. (2018). La notation est adaptée au contexte de ce mémoire.

Expression	**Définition**
$\phi \in [0, 1]$	Valeur d'une fidélité
$f(x, \phi)$	Valeur de l'objectif suite à une évaluation au point x et à une fidélité ϕ
$\lambda^c : [0, 1] \to \mathbb{R}^+$	Fonction coût monotone croissante selon ϕ

Tableau 2.2 Définitions liées à la multi-fidélité.

La plus haute fidélité correspond à $\phi = 1$, alors que $\phi = 0$ est la plus basse. Ce que ces fidélités signifient concrètement pour un procédé doit être établi par l'utilisateur. Pour faire suite à l'exemple de la simulation Monte-Carlo donné plus haut, $\phi = 1$ pourrait correspondre à $2 \cdot 10^5$ tirages alors que $\phi = 0$ pourrait correspondre à 100 tirages. Ces correspondances dépendent du problème auquel fait face l'utilisateur. Alarie et al. (2021b) proposent une méthode qui ajuste la précision d'une boîte noire sans contraintes au fur et à mesure de l'optimisation et qui s'applique bien aux simulations MC. La troisième définition du tableau 2.2 impose que λ^c augmente avec la fidélité. Cette définition sera utile pour garantir l'optimalité d'un sous-problème d'optimisation détaillé à la section 4, où une diminution de la fidélité doit impliquer une diminution du temps d'évaluation. La variable λ^c est définie comme le coût à des fins de généralité. Or, pour ce projet de maîtrise, ce coût correspond au temps d'évaluation.

En BBO, une boîte noire peut être multi-fidélité. Le cas échéant, à chaque évaluation de la boîte noire, celle-ci doit recevoir en entrée non-seulement un point $x \in X$, mais également une valeur de fidélité $\phi \in [0, 1]$ avec laquelle effectuer l'évaluation. En ce qui concerne λ^c, plusieurs champs de la recherche opérationnelle assument que cette fonction est connue. Toutefois, comme les méthodes d'optimisation de boîtes noires ont pour but de considérer un cadre très général, les hypothèses posées sur λ^c sont les plus faibles possible. De ce fait, mis à part la monotonie donnée par la définition du tableau 2.2, aucune hypothèse n'est posée sur λ^c.

Pour lister quelques nouveaux exemples, une boîte noire multi-fidélité peut être constituée d'une simulation par éléments finis où la taille du maillage dicte la fidélité, ou d'un algorithme d'apprentissage machine dont on veut optimiser les hyperparamètres et où la fidélité est donnée par le nombre d'itérations d'apprentissage tel que montré par Wu et al. (2020). Pareillement, il pourrait s'agir d'une expérience en laboratoire où une fidélité plus basse est atteinte en se « dépêchant » d'effectuer les manipulations au risque d'effectuer des erreurs. Ce dernier exemple est plutôt excentrique, mais il illustre à quel point l'optimisation de boîtes noires multi-fidélité est large par sa nature très générale. Bien sûr, toute boîte noire multi-fidélité peut être vue comme une boîte noire qui ne prend que x comme paramètre en fixant la fidélité à priori.

Dans la littérature récente, Belakaria et al. (2020) développent une méthode pour approximer un front de Pareto en multi-objectif multi-fidélité. Aussi, Wang et al. (2022) proposent une extension à l'algorithme d'arbre de recherche optimiste hiérarchique (*hierarchical optimistic tree search*) pour utiliser les évaluations à basse fidélité.

2.4 Boîtes noires stochastiques

La multi-fidélité est souvent le produit d'un processus stochastique. En revanche, les méthodes d'optimisation de boîtes noires stochastiques s'appliquent très différemment des méthodes d'optimisation de boîtes noires multi-fidélité. En optimisation stochastique, il n'est généralement pas assumé que le contrôle sur la précision existe, seulement que les sorties d'une boîte noire sont bruitées. En optimisation multi-fidélité, la précision est contrôlable par un ou plusieurs paramètres mappés sur la fidélité qui appartient à $[0, 1]$. Les approches sont donc plutôt distinctes.

Tel que mentionné plus tôt, ce travail de maîtrise testera la méthode proposée avec une implémentation de l'algorithme MADS, pour lequel des adaptations pour les boîtes noires stochastiques ont été développées récemment. Par exemple, Audet et al. (2018b) proposent

une version de MADS nommé Robust-MADS qui construit une fonction lisse à partir des évaluations bruitées pour approximer l'objectif. Cette méthode aborde le problème des boîtes noires contaminées par un bruit numérique. Aussi, Audet et al. (2021) proposent StoMADS, une autre version de MADS, plutôt adaptée à un cadre stochastique, en imposant un seuil sur la probabilité que l'estimé d'une évaluation soit précise et une condition sur la variance de ces évaluations. En opposition aux autres méthodes qui ne posent aucune hypothèse sur le bruit, Alarie et al. (2021b) suggèrent une modification à MADS qui assume un bruit gaussien. L'algorithme proposé fait tendre vers zéro l'écart-type des estimations des valeurs de l'objectif, tout en conservant plusieurs propriétés de convergence.

L'algorithme de Nelder-Mead a également vu plusieurs modifications pour être applicable à des systèmes stochastiques. Parmi les premiers à suggérer de telles modifications, Barton et Ivey, Jr. (1996) proposent d'ajouter des réévaluations de l'itéré courant, et une atténuation de l'étape de contraction. Aussi, Anderson et Ferris (2001) proposent un nouvel algorithme de BBO très semblable à Nelder-Mead, où les simplexes sont remplacés par d'autres structures plus adaptées dans un contexte stochastique. Plus récemment, Chang (2012) propose de remplacer l'étape de réduction de Nelder-Mead par une recherche aléatoire adaptative.

2.5 PRIAD

PRIAD est le projet d'Hydro-Québec qui a motivé ce projet de recherche. Le simulateur qui sera vu comme une boîte noire est décrit par Komljenovic et al. (2019). PRIAD y est présenté comme une approche qui vise à maximiser la valeur des actifs d'Hydro-Québec, leur durabilité et leur résilience. Ces aspects sont abordés d'une part d'un point de vue technique, et d'autre part d'un point de vue organisationnel dans l'entreprise. Le simulateur d'intérêt pour ce projet est constitué de quatre modules appelés séquentiellement, illustrés à la figure 2.3.

D'abord, l'optimiseur envoie un point à la boîte noire. Ce dernier correspond aux périodicités de maintenance de chaque type de maintenance pour chaque famille d'équipement. L'ensemble des familles d'équipement est de taille N, et comprend les transformateurs, les disjoncteurs, etc. Il est également possible de diviser une famille en plusieurs sous-familles, si par exemple il est jugé pertinent d'attribuer différentes périodicités aux transformateurs de basse puissance, de moyenne puissance et de haute puissance. Le premier module, qui est détaillé par Côté et al. (2020), modélise les mécanismes de dégradation selon les périodicités données, pour produire un taux de défaillance λ pour chaque famille d'équipement. Ce module utilise un modèle basé sur la physique théorique et les connaissances des experts du domaine pour prédire de premiers taux de défaillances. Ce modèle est alors comparé à des données historiques pour calibrer le modèle. Cette calibration est effectuée par un modèle d'optimisation

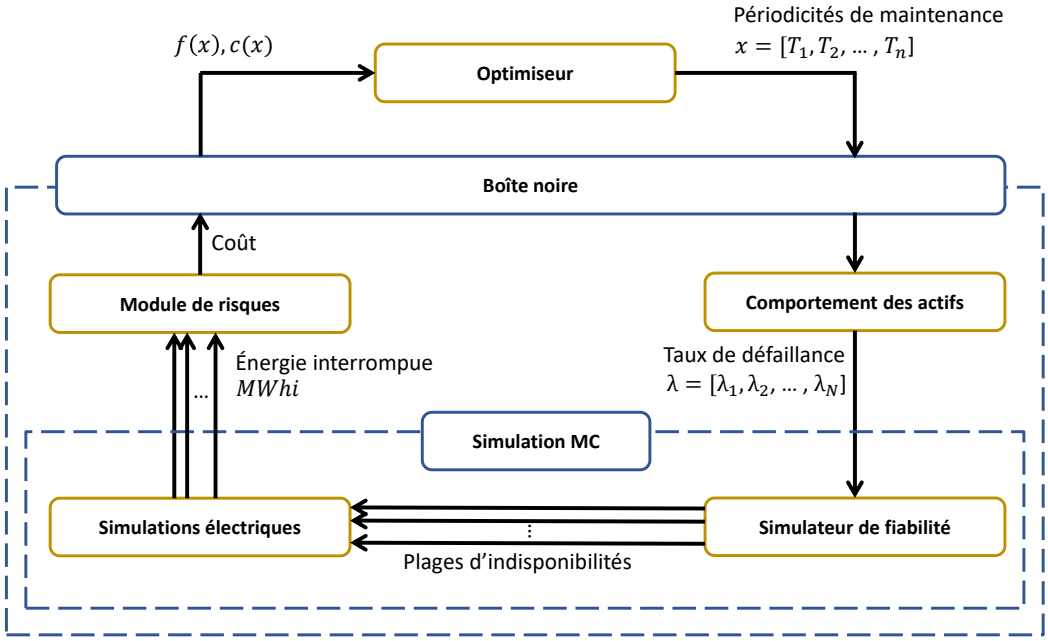


FIGURE 2.3 Représentation graphique du simulateur de PRIAD.

analytique. Les taux de défaillance, qui sont des probabilités que chaque équipement encoure une défaillance chaque année, permettent au second module d'effectuer plusieurs simulations de fiabilités. Chaque simulation correspond à un tirage par la méthode MC, et renvoie une plage d'indisponibilités de chaque équipement individuel sur un horizon de 40 ans. Gaha et al. (2021) détaillent davantage les deux derniers modules. Pour chacun des tirages MC, l'avant-dernier module effectue une simulation de l'écoulement de puissance optimal, compte tenu du fait que certains équipements sont indisponibles durant certaines périodes. De plus, l'opération du réseau, c'est-à-dire, entre autres, les changements d'états de disjoncteurs ou de sectionneurs en cas de défaillances ou de maintenance planifiée respectivement, est également simulée. Chaque simulation renvoie l'énergie non livrée sur le réseau, indiquée par l'acronyme $MWhi$ qui signifie Mégawatts heure interrompus. Finalement, le module de risque traduit les sorties des différents modules en termes monétaires. Ce module prend en compte, entre autres, le coût de la maintenance planifiée, le nombre de pannes concourantes, le coût de l'énergie non livrée et l'impact des défaillances sur les infrastructures touchées. Par exemple, un grand coût est associé à une simulation où un hôpital est dépourvu d'électricité pendant une grande durée.

Komljenovic et al. (2019) montrent également le tableau 2.3. Ce dernier présente des estimations du temps qu'une seule évaluation de la boîte noire requiert. Ces temps sont obtenus

en supposant qu'aucune parallélisation n'est effectuée, et qu'un nombre limité d'équipements est simulé (il y a beaucoup plus que 2000 équipements électriques sur le réseau de transport d'Hydro-Québec).

Cas	**Nombre d'équipements**	**Nombre de tirages MC**	**Nombre d'évènements simulés**	**Temps d'une évaluation (jours)**
Sous-station	80	2×10^4	5.6×10^6	0.23
Corridor	400	1×10^5	1.4×10^8	5.83
Réseau	2000	5×10^5	3.5×10^6	145

TABLEAU 2.3 Temps d'évaluation du simulateur de PRIAD par Komljenovic et al. (2019).

Dans un cadre général, Murphy et al. (2005) décrivent comment déterminer ce qui constitue la vérité dans le domaine d'étude des simulations de fiabilité, en comparant des approches théoriques et expérimentales. Aussi, Murphy et al. (2001) indiquent quelques règles générales à suivre pour conduire différents types de simulations de fiabilité.

2.6 Réduction du temps d'optimisation

En pratique, les méthodes de BBO sont souvent appliquées à des boîtes noires coûteuses à évaluer. Cette section présente quelques-uns des travaux qui visent la réduction du temps total d'optimisation. Razavi et al. (2010) divisent les types d'approches en quatre catégories :

- Le développement d'algorithmes particulièrement pour les problèmes coûteux. Le développement de NOMAD tombe dans cette catégorie ;
- L'utilisation du parallélisme pour lancer plusieurs simulations simultanément. Cette avenue a été explorée par Audet et al. (2008) et Alarie et al. (2018) dans le contexte du logiciel NOMAD ;
- L'identification des évaluations à simplement éviter. La méthode proposée dans ce mémoire tombe dans cette catégorie ;
- L'utilisation de substituts et modèles peu coûteux qui imitent la boîte noire, tel qu'abordés par Conn et al. (1997).

Du côté des algorithmes plus efficaces, Huot et al. (2019) suggèrent une approche qui combine MADS à l'algorithme de recherche dimensionné dynamiquement : *Dynamically Dimensioned Search* (DDS) pour former un nouvel algorithme hybride. Ce dernier a été testé sur la

calibration de modèles hydrologiques et une amélioration significative du temps d'optimisation a été observée. Aussi, Wetter et Polak (2005) proposent une méthode qui s'apparente aux modèles pour résoudre des systèmes d'équations différentielles complexes. La méthode débute par effectuer de nombreuses approximations pour explorer le domaine des variables, puis la précision, et donc le temps d'évaluation aussi, sont augmentés graduellement. Les premières itérations sauvent suffisamment de temps pour observer une réduction significative du temps total pour les systèmes d'équations testés.

Finalement, en ce qui concerne l'évitement des mauvaises évaluations, Alarie et al. (2022) proposent deux algorithmes de hiérarchisation d'une boîte noire sous contraintes. La publication s'intéresse au cas où au cours d'une évaluation, différentes étapes donnent différentes informations intermédiaires. Il est alors possible de poser des contraintes sur ces informations intermédiaires. La boîte noire peut ainsi être considérée comme un ensemble de sous-boîtes noires, où chacune renvoie la valeur d'une contrainte, sauf une qui donne la valeur de l'objectif. Le premier algorithme est une variation sur la BE. Durant la phase de réalisabilité, la fonction de violation des contraintes est calculée après chaque sous-boîte noire, en ne considérant que les contraintes évaluées. Dès que cette fonction atteint une valeur plus grande que celle de l'itéré courant, l'évaluation est interrompue. Durant la phase d'optimalité, dès qu'une sous-boîte noire renvoie une contrainte qui n'est pas satisfaite, l'évaluation est interrompue. Du temps est alors sauvé sur les évaluations de mauvaise qualité. Le second algorithme repose sur une hiérarchie des sous-boîtes noires. D'abord, les contraintes sont classées de la plus difficile à satisfaire à la plus facile. L'algorithme comporte également une phase de réalisabilité, où la valeur de la première contrainte est minimisée, sans contraintes. Une fois qu'un point est tel que l'objectif est nul (la première contrainte est satisfaite), un second problème est résolu, où la première contrainte est la seule contrainte et la seconde est minimisée. À chaque fois qu'un problème est résolu, la contrainte qui était minimisée est traitée comme une contrainte et une nouvelle contrainte à minimiser s'ajoute dans le prochain problème. Un fois la phase d'optimalité atteinte, le problème redevient le problème original en ajoutant l'objectif, et le principe des interruptions du premier algorithme est appliqué. Cette approche permet de traiter en priorité les contraintes particulièrement difficiles à satisfaire qui peuvent faire perdre beaucoup de temps à un algorithme. Ces deux algorithmes ont directement inspiré la méthode présentée dans ce mémoire.

2.7 Théorie des graphes

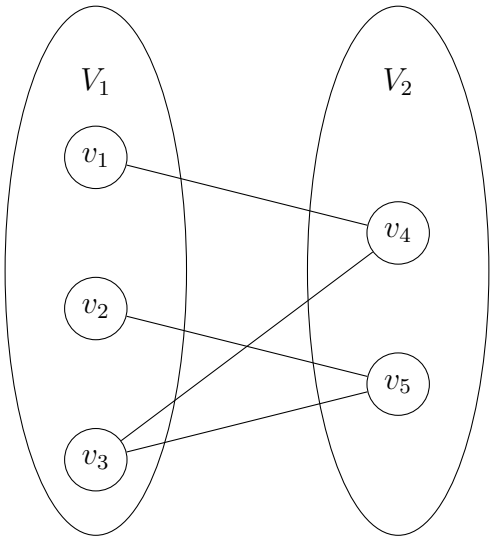
La théorie des graphes est une branche des mathématiques qui offre une manière intéressante de modéliser des problèmes d'optimisation. Il est important de mentionner qu'aucun

algorithme d'optimisation en lien avec la théorie des graphes n'est utilisé dans ce mémoire. Plutôt, le système de notation et le vocabulaire de la théorie des graphes sont utilisés pour simplifier l'écriture. De plus, ce domaine permet d'illustrer certains concepts abstraits avec une grande clareté.

Un graphe G est définit par deux ensembles. Un premier ensemble V contenant les sommets, et un second ensemble nommé E contenant les arêtes qui relient les sommets entre eux. Une arête reliant les sommets v_1 et v_2 est notée $[v_1, v_2]$. Le degré d'un sommet v noté $\deg(v)$ est le nombre d'arêtes incidentes à ce sommet. Un graphe est biparti s'il existe une partition de son emsemble V en deux ensembles V_1 et V_2 telle que chaque arête de G a une extrémité dans V_1 et l'autre dans V_2. La matrice d'adjacence A d'un graphe est une matrice de dimension $|V|^2$ où l'élément A_{ij} est égal à 1 s'il existe une arête reliant le i-ième et le j-ième sommet, et est égal à 0 sinon. La matrice d'adjacence d'un graphe biparti contient beaucoup d'information redondante, et peut donc être écrite avec la forme

$$A = \begin{bmatrix} 0 & B \\ B^\top & 0 \end{bmatrix},$$

où les 0 sont des matrices ne contenant que des zéros. La matrice B se nomme la matrice de biadjacence. La figure 2.4 illustre ces propos avec un exemple. Dans ce dernier, les degrés des sommets v_1 à v_5 sont respectivement 1, 1, 2, 2 et 2.



$$A = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{array}\right]$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

FIGURE 2.4 Exemple de graphe biparti avec ses matrices d'adjacence et de biadjacence.

CHAPITRE 3 SOUS-ÉVALUATIONS INTERROMPUES

Ce chapitre vise à montrer comment la méthode proposée réduit le temps d'optimisation d'une boîte noire coûteuse. La première section de ce chapitre établit certaines définitions qui sont utilisées dans le reste de ce mémoire. La seconde section décrit la méthode proposée, appelée la méthode d'optimisation avec contraintes hiérarchisées. Finalement, la dernière section décrit en détails un algorithme qui fait partie de la méthode d'optimisation avec contraintes hiérarchisées

3.1 Notation et définitions

Comme la méthode proposée est originale, des nouvelles définitions sont données au tableau 3.1.

Expression	Définition
$c(x, \phi) \in \mathbb{R}^m \cup \{\infty\}^m$	Vecteur des contraintes suite à une évaluation au point x et à une fidélité ϕ
$t(x, \phi) \in \mathbb{R}_{\geq 0}$	Temps d'évaluation au point x et à une fidélité ϕ, coût monotone croissant selon ϕ
$\Phi \in [0, 1]^\ell$	Ensemble de ℓ fidélités choisies par l'utilisateur
$\phi_i \in \Phi$	i-ième fidélité de Φ, où $i \in I := \{1, 2, \ldots, \ell\}$

TABLEAU 3.1 Définitions liées à l'algorithme de sous-évaluations interrompues.

La première définition du tableau 3.1 est une adaptation aux contraintes de la première définition du tableau 2.2. En effet, les méthodes d'optimisation de boîte noire multi-fidélité ou d'optimisation stochastique présentées à la section 2 considèrent uniquement le cas non contraint. Seulement la valeur de l'objectif en fonction de x et de ϕ a donc été définie. La seconde définition est une généralisation de la dernière définition du tableau 2.2 au cas où le coût peut varier avec les paramètres d'entrée en plus de la fidélité. Aussi, pour ce projet de maîtrise, le seul coût associé à une évaluation est le temps. Pour tout point $x \in X$, $t(x, \phi)$ est une fonction monotonne croissante selon ϕ. Si la multi-fidélité est le résultat d'un

processus stochastique, il est possible que $c(x, \phi)$ et $t(x, \phi)$ varient d'une évaluation à l'autre pour le même x et le même ϕ. Dans ce cas, $c(\cdot)$ et $t(\cdot)$ ne sont pas des fonctions. Ce constat est également valide pour $f(x, \phi)$. Concernant les deux dernières définitions du tableau 3.1, comme $\phi \in [0, 1]$, il existe une quantité infinie de fidélités auxquelles il est possible d'appeler une boîte noire. La méthode présentée dans ce mémoire ne peut que considérer une quantité finie de fidélités, l'utilisateur doit alors choisir un ensemble Φ où $\ell \neq \infty$, qui doit inclure 1. Le choix de la taille de Φ dépend de la capacité paralléle de l'utilisateur. Cet aspect est détaillé dans un prochain chapitre.

Étant donné un problème d'optimisation de boîte noire, la méthode d'optimisation avec contraintes hiérarchisées construit un graphe biparti $G = (V_1, V_2, E)$. Dans ce graphe, chaque fidélité $\phi_i \in \Phi$ est représentée par un sommet, et ces fidélités forment la première partie de la partition : $V_1 = \{\phi_1, \phi_2, \ldots, \phi_\ell\} = \Phi$. Chaque contrainte est également représentée par un sommet, et ces contraintes forment la seconde partie : $V_2 = \{c_1, c_2, \ldots, c_m\}$. Durant l'exécution de la méthode, les contraintes sont assignées à des fidélités. Cela est représenté par l'ensemble E, qui est tel que

$$\{[c_j, \phi_i], [\phi_i, c_j]\} \subseteq E \iff c_j \text{ est assignée à } \phi_i.$$

Chaque contrainte est assignée à exactement une fidélité, ce qui signifie que le degré de chaque sommet de V_2 doit être égal à un. Plus une contrainte est assignée à une grande fidélité, plus elle est élevée dans la hiérarchie des contraintes. L'ensemble E est donné par la matrice de biadjacence B. La matrice d'adjacence de G n'est pas utilisée car elle est de taille $(\ell + m) \times (\ell + m)$, alors que la matrice B contient la même information tout en étant de taille $\ell \times m$.

3.2 Algorithme d'optimisation avec contraintes hiérarchisées

La méthode proposée consiste à aissgner les contraintes d'un problème d'optimisation de boîte noire à des fidélités. L'objectif de ces assignations est de réfleter à partir de quelle fidélité chaque contrainte donne de l'information pertinente, où $\phi = 1$ est la vérité, donc l'information la plus pertinente. Cela peut permettre d'effectuer des appels à la boîte noire à des fidélités plus basses (qui sont plus rapides à exécuter) pour déterminer si certaines contraintes ne sont pas satisfaites. Le cas échéant, l'algorithme est interrompu. Plus cette terminaison survient tôt, plus il y a de temps sauvé. L'idée derrière cette méthode est qu'un point non réalisable n'est pas intéressant, et qu'une évaluation coûteuse avec un tel point est une perte de temps à éviter. Plus concrètement, la méthode est composée de trois étapes

montrées à l'algorithme 3.1.

Algorithme 3.1 : Optimisation avec contraintes hiérarchisées

Étant donné un problème d'optimisation donné par une boîte noire multi-fidélité

1. Analyser le comportement des contraintes en fonction de la fidélité
2. Déterminer une matrice de biadjacence optimale
3. Lancer l'optimisation avec un solveur

Il est nécessaire que l'ensemble E soit déterminé en fonction du comportement des contraintes à différentes fidélités pour qu'il permette des interruptions justes. De ce fait, la première étape consiste à effectuer une analyse de ces comportements. La seconde étape se base sur cette analyse pour calculer une matrice de biadjacence optimale. Sachant que cette matrice contient l'information sur les arêtes, elle est utilisée pour contenir l'information sur les assignations des contraintes. Une matrice de biadjacence est optimale si elle minimise l'espérance du temps d'évaluation. Après l'étape deux, le graphe G est défini à partir de V et E. Finalement, à la dernière étape, l'optimisation est lancée. Cette optimisation est effectuée par un solveur existant. Toutefois, au lieu de donner au solveur la boîte noire directement, une passerelle est donnée. Celle-ci constitue une implémentation d'un algorithme qui lance la boîte noire séquentiellement aux différentes fidélités en suivant la hiérarchie. À chaque appel de la boîte noire, seulement les contraintes adjacentes à la fidélité courante ou à une fidélité inférieure dans G sont considérées. Après chaque appel, si l'une de ces contraintes n'est pas satisfaite, l'algorithme se termine. Si le graphe G reflète bien quelles contraintes ont des valeurs pertinentes à quelles fidélités et que la boîte noire le permet, les interruptions identifieront des points non réalisables sans avoir besoin d'obtenir la vérité.

3.3 Algorithme de sous-évaluations interrompues

L'algorithme de sous-évaluations interrompues est l'algorithme implémenté dans la passerelle. À chaque fois qu'elle reçoit un point du solveur (ou qu'elle reçoit le point de départ), l'algorithme de sous-évaluations interrompues est lancé, qui lui effectue les appels à la boîte noire en considérant l'aspect multi-fidélité. Cette relation est illustrée à la figure 3.1. Cette figure reprend le schéma de la figure 2.1 pour visualiser l'insertion de la passerelle, qui lance séquentiellement la boîte noire à différentes fidélités. Ces lancements sont appelés des sous-évaluations, de sorte que le terme évaluation soit relatif au point de vue du solveur. Cela

permet de rester cohérent avec la littérature. Chaque évaluation constitue un achèvement de l'algorithme de sous-évaluations interrompues, qui lui effectue plusieurs sous-évaluations.

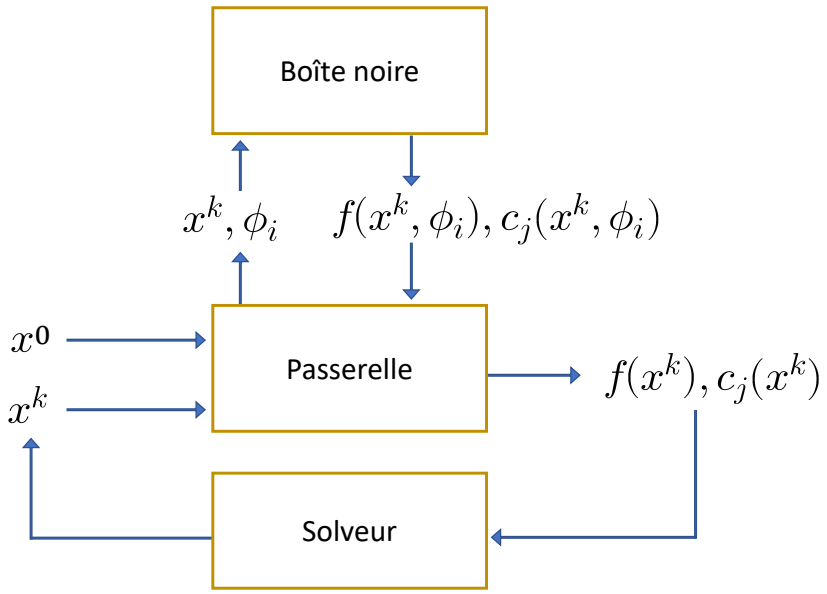


FIGURE 3.1 Boucle d'optimisation avec un solveur et la passerelle.

Cet algorithme est détaillé à l'algorithme 3.2, sous forme de fonction. Comme la distinction entre les différents x^k n'est pas pertinente pour l'algorithme, seulement x est utilisé pour dénoter le point en évaluation. La fonction *Sous-évaluation* : $(\mathbb{R}^n, [0,1]) \to (\mathbb{R} \cup \{\infty\}, (\mathbb{R} \cup \{\infty\})^m)$ dénote un lancement de la boîte noire, et elle renvoie $f(x, \phi)$ et $c(x, \phi)$. Les expressions e_i et $\mathbf{0}$ dénotent respectivement le vecteur de longueur m qui ne contient que des 0 à l'exception du i-ième élément qui vaut 1 et le vecteur de longueur m qui ne contient que des 0. Le paramètre f^* dénote la valeur de la fonction objectif à la meilleure solution réalisable trouvée depuis le début de l'optimisation. Cette valeur est obtenue en lançant une fonction nommée *Obtenir-meilleure-valeur*. Cette dernière pourrais, par exemple, lire un fichier pour renvoyer f^*.

Pour chaque fidélité, en ordre croissant (d'où le principe de hiérarchie), l'algorithme débute par vérifier si $e_i^\top B \neq \mathbf{0}$. Ce test vérifie qu'au moins une contrainte est assignée à ϕ_i. Si ce n'est pas le cas, l'algorithme passe à la prochaine fidélité. Autrement, une sous-évaluation

Algorithme 3.2 : Sous-évaluations interrompues

Fonction *Évaluation*$(x, \textit{Sous-évaluation}(\cdot), G)$

Déclarer f, c

$f^* \leftarrow \textit{Obtenir-meilleure-valeur}()$

Pour tout $\phi_i \in \Phi$

  Si $e_i^\top B \neq \mathbf{0}$

    $f, c \leftarrow \textit{Sous-évaluation}(x, \phi_i)$

    Pour tout $j \in J$ tel que $\{B_{aj} \in B : B_{aj} = 1, a \leq i\} \neq \varnothing$

      Si $c_j > 0$

        Interruption, l'algorithme se termine

        Renvoyer f, c

Si $e_\ell^\top B = \mathbf{0}$ et $f < f^*$

  $f, c \leftarrow \textit{Sous-évaluation}(x, 1)$

Renvoyer f, c

est effectuée. Par la suite, une boucle itère sur les j qui sont tels que $\{B_{aj} \in B : B_{aj} = 1, a \leq i\} \neq \varnothing$. Cette condition vérifie que c_j est une contrainte assignée ($B_{aj} = 1$) à la fidélité courante ou à une fidélité inférieure ($a \leq i$). Si ces contraintes ne sont pas satisfaites, l'algorithme se termine. Autrement, la boucle principale se poursuit. Si cette boucle se termine sans interruption, cela signifie que $c(x, \phi) \leq \mathbf{0}$ pour la plus grande fidélité dans Φ à laquelle est assignée au moins une contrainte. Or, il est possible que cette fidélité ne soit pas 1, si $e_\ell^\top B = \mathbf{0}$ (sachant que $\phi_\ell = 1$). De plus, si l'objectif à la dernière sous-évaluation est meilleur que l'objectif au meilleur point trouvé depuis le début de l'optimisation, une sous-évaluation supplémentaire est ajoutée à la fidélité correspondant à la vérité ($\phi = 1$). L'intérêt de cette pratique est que l'objectif à cette dernière sous-évaluation peut être différent de l'objectif à $\phi = 1$ pour le même point. La sous-évaluation ajoutée à la fin assure que la valeur de f renvoyée au solveur est la vraie si le solveur s'apprête à définir une nouvelle meilleure solution. De plus, il est possible que x soit réalisable à la dernière sous-évaluation, mais ne le soit pas à $\phi = 1$. Cela est rare si B est de qualité (reflète bien le comportement des contraintes), mais il est prioritaire de garantir qu'un point qui pourrait être renvoyé à l'utilisateur en tant que meilleure solution réalisable si le solveur se termine soit véritablement réalisable.

Pour illustrer ces propos, la figure 3.2 montre un exemple de problème où $m = 5$ et $\ell = 4$.

En lançant l'algorithme 3.2 avec ce graphe, la boîte noire est d'abord évaluée à ϕ_1, fidélité à laquelle sont assignées les contraintes c_1 et c_2. Si ces contraintes ne sont souvent pas satisfaites à ϕ_1, plusieurs interruptions auront lieu après la première évaluation durant l'optimisation. Si ces contraintes sont également non satisfaites si une évaluation à $\phi = 1$ était effectuée, les interruptions sont justifiées. Sachant qu'une évaluation à ϕ_1 est la moins coûteuse parmi les $\phi \in \Phi$ pour un même x, beaucoup de temps est sauvé avec ces interruptions durant une optimisation. Toujours dans le même exemple, aucune contrainte n'est partitionnée à ϕ_2 car $e_2^\top B = [0, 0, 0, 0, 0]$. L'algorithme ne lance donc pas de sous-évaluation avec ϕ_2. Si $c_1(x, \phi_1)$ et $c_2(x, \phi_1)$ sont satisfaites, l'algorithme effectue la prochaine sous-évaluation à ϕ_3. Cette fois, les contraintes c_1, c_2, c_3 et c_4 sont vérifiées. La raison pour laquelle les contraintes assignées plus basses dans la hiérarchie sont encore vérifiées est que leur valeur peut changer en augmentant la fidélité, et ces nouvelles valeurs sont probablement plus proches de la vérité. De plus, il n'y a aucun coût associé à simplement vérifier une contrainte. Les contraintes c_3, c_4 et c_5 sont assignées à des fidélités plus coûteuses que c_1 et c_2. Elles ne permettront pas de sauver autant de temps. De plus, si les c_j vérifiées sont toutes satisfaites à toutes les sous-évaluations avant ϕ_4, l'achèvement de l'algorithme prendra plus de temps que si une seule sous-évaluation à ϕ_4 était directement effectuée. En revanche, si c_4 ne donne de l'information pertinente qu'à ϕ_4, assigner c_4 à des plus basses fidélités peut mener à des interruptions injustifiées, c'est-à-dire des interruptions où un point est réalisable en vérité. Conséquemment, une matrice de biadjacence est de qualité si elle permet d'identifier avec justesse si certaines contraintes sont satisfaites ou non en vérité à partir d'information à différentes fidélités. Aussi, la quantité de temps qu'il est possible de sauver pour un problème d'optimisation donné dépend du comportement des contraintes. Si une boîte noire est telle que les fidélités inférieures à 1 n'offrent pas d'information pertinente au sujet des contraintes, il n'y a pas d'interruptions pertinentes à effectuer.

Une sous-évaluation a été définie précédemment comme le lancement d'une boîte noire par la passerelle. Il est toutefois possible d'élargir cette définition pour tenir compte du cas où une boîte noire donne de l'information au fur et à mesure de son exécution. Par exemple, dans le cas de PRIAD, la simulation peut renvoyer des sorties après l'exécution du premier module, après l'exécution du second module, après certaines quantités de tirages MC, et après le dernier module. Chaque fidélité correspond à un moment durant la simulation où des sorties sont disponibles. Dans ce contexte, à chaque évaluation, la passerelle ne lance la boîte noire qu'une seule fois, et elle recevra des sorties pour chaque fidélité durant l'exécution de la boîte noire. Une interruption correspond à un arrêt de la boîte noire durant son exécution. Finalement, une sous-évaluation à la fidélité ϕ_i correspond au fonctionnement de la boîte noire de son lancement jusqu'à l'atteinte des sorties qui correspondent à ϕ_i. La méthode

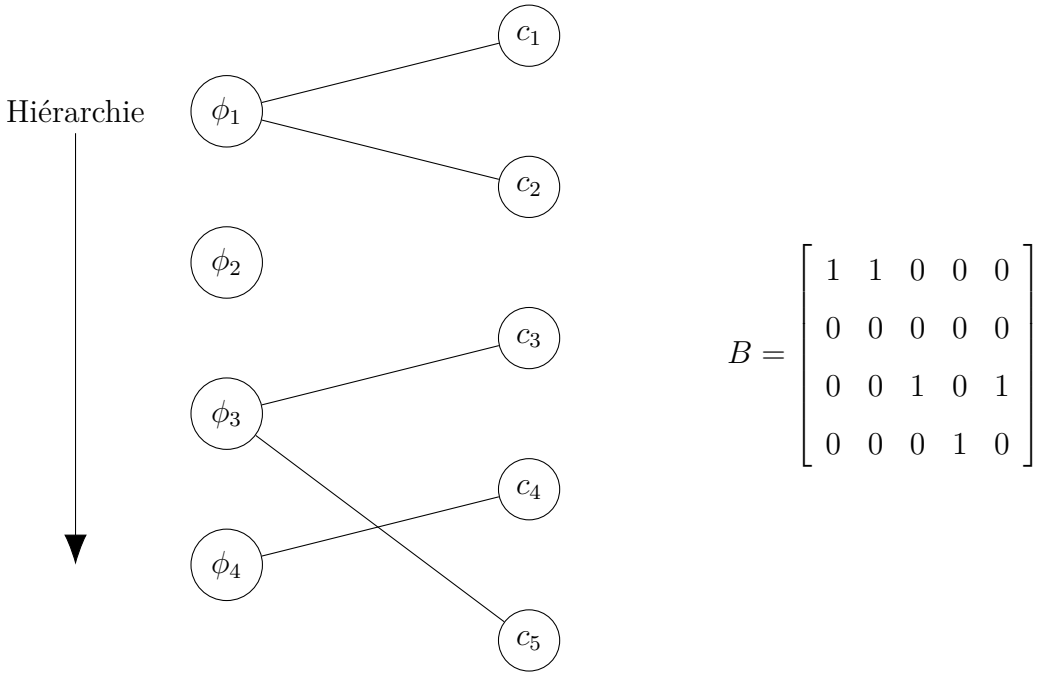


FIGURE 3.2 Exemple de graphe avec sa matrice de biadjacence.

présentée n'est pas affectée par ce cas, seulement l'implémentation logicielle doit être adaptée. L'avantage d'une telle pratique est qu'il est plus facile de sauver du temps. Une évaluation prend au pire le temps d'une sous-évaluation à ϕ_ℓ, et au mieux le temps d'une sous-évaluation à ϕ_1. Dans le cas plus fréquent où une boîte noire ne donne ses sorties qu'à la fin de son exécution, une évaluation prend au pire un temps égal à la somme des temps des sous-évaluations, et au mieux le temps d'une sous-évaluation à ϕ_1.

Pour terminer, maintenant que la méthode a été décrite, il est possible de lister les propriétés qu'une boîte noire doit posséder pour que la méthode soit applicable.

— La possibilité de lancer la boîte noire à différentes fidélités.
— L'existence de contraintes qui donnent de l'information pertinente à des fidélité inférieures à 1.
— L'existence d'optimum locaux près de ces contraintes.

Plus il y a de contraintes difficiles à satisfaire, plus une méthode qui ne fait que des évaluations à fidélité maximale passe de temps sur des points non réalisables. Plus ces contraintes donnent de l'information pertinente à des fidélités basses, plus l'algorithme 3.2 effectuera des interruptions qui résulteront en un meilleur temps. Toutefois, si le solveur converge vers un optimum qui est à la limite de la réalisabilité d'une contrainte qui ne donne de l'information pertinente qu'à grande fidélité (ou s'il converge vers un optimum qui n'est pas près du contour de Ω), il est possible que les interruptions soient rares. Conséquemment, plus les contraintes

qui donnent de l'information utile à basse fidélité sont celles qui définissent les contours de Ω proches desquelles se situent des optimums locaux parmi les points de Ω, plus il y a de temps à réduire.

CHAPITRE 4 CALCUL DE LA MATRICE DE BIADJACENCE

Le quatrième chapitre détaille les deux premières étapes de l'algorithme 3.1. Avant de lancer une optimisation sur une boîte noire, certaines démarches sont effectuées pour obtenir une matrice de biadjacence (des assignations des contraintes aux fidélités) de qualité. Il y a d'abord une analyse du comportement des contraintes, puis la création d'un modèle d'optimisation analytique qui détermine une matrice de biadjacence optimale. Ce modèle ne correspond pas exactement au problème ; les différences sont expliquées et justifiées. Aussi, le modèle est long et difficile à résoudre. Une méthode de résolution rapide et adaptée spécifiquement à ce modèle d'optimisation est présentée.

4.1 Analyse du comportement des contraintes

Cette section présente la première étape de l'algorithme 3.1. L'analyse de l'impact de la fidélité sur les contraintes est effectuée à l'aide d'un échantillon de points. Cet échantillon est obtenu par un hypercube latin. Chaque point de l'hypercube latin est évalué à chacune des fidélités de Φ. L'avantage de cette pratique est que chaque évaluation est une tâche indépendante, ce qui signifie qu'il est possible de les lancer parallèlement. L'utilisateur doit choisir une taille de Φ ainsi qu'une taille d'hypercube latin en conséquence de sa capacité à lancer des tâches informatiques en parallèle. Plus le nombre de fidélités ℓ et la taille de l'échantillon sont grands, plus la matrice B calculée par la suite sera de qualité. Or, il doit être considéré que l'objectif de la méthode est de réduire le temps d'optimisation, donc il est favorable de passer peu de temps sur cette première étape. Quoi qu'il en soit, la fidélité correspondant à la vérité ($\phi = 1$) doit appartenir à Φ pour l'analyse.

D'abord, les bornes de l'hypercube latin doivent être déterminées. Il est possible d'échantilloner l'entièreté de l'espace des paramètres si les bornes de l'hypercube latin sont définies par les vecteurs $l \in \mathbb{R}^n$ et $u \in \mathbb{R}^n$ (les bornes de X). Dans le contexte de ce projet de maîtrise, on souhaite obtenir de l'information sur les points qui risquent d'être évalués lors de l'optimisation. Il peut alors être préférable d'évaluer un hypercube latin centré sur le point de départ de l'optimisation, et avec des bornes plus rapprochées du point de départ. On définit un nouveau paramètre $\Delta \in]0, 2]$, qui est le facteur par lequel chaque borne est réduite. Une tranlsation est également effectuée sur les bornes pour placer le point de départ x^0 au centre des bornes inférieures et supérieures. Si une borne est à l'extérieur de X après une translation, elle est redéfinie par la borne de X. Ces nouvelles bornes sont nommées l^{cen} et u^{cen}, et

elles sont calculées par

$$l_i^{\text{cen}} = \max\left(l_i, x_i^0 - \frac{\Delta}{2}(u_i - l_i)\right) \quad \forall\, i \in I$$
$$u_i^{\text{cen}} = \min\left(u_i, x_i^0 + \frac{\Delta}{2}(u_i - l_i)\right) \quad \forall\, i \in I.$$

Il est à noter que l'utilisateur peut choisir $\Delta = 2$ pour échantillonner sur X en entier.

Pour une boîte noire comme celle de PRIAD où pendant une même évaluation, de l'information sur plusieurs fidélités est disponible, il n'est pas nécessaire d'évaluer un hypercube latin. La première étape de l'algorithme 3.1 peut simplement être constituée des premières évaluations d'une optimisation où chaque évaluation est un appel de la boîte noire avec $\phi = 1$. Durant ces évaluations, les sorties à toutes les fidélités de Φ et à tous les pionts sont collectées. La matrice de biadjacence peut ensuite être calculée à partir de ces informations et à la dernière étape de l'algorithme 3.1, il suffit de poursuivre l'optimisation déjà entamée.

Pour l'algorithme de sous-évaluations interrompues (algorithme 3.2), la valeur exacte des contraintes n'est pas importante. Elle est gardée en mémoire seulement pour être renvoyée au solveur qui lui peut s'en servir. L'important est de vérifier si une contrainte est satisfaite ou non, pour décider si une interruption doit avoir lieu ou non. La fonction binaire suivante est alors définie, et est utilisée dans plusieurs autres définitions.

$$\beta(c_j(x,\phi)) := \begin{cases} 0 & \text{si } c_j(x,\phi) \leq 0 \\ 1 & \text{sinon} \end{cases} \qquad \forall j \in J.$$

Soit une contrainte c_j avec $j \in J$, un point $x \in X$ et une fidélité $\phi_1 \in \Phi$. La fidélité ϕ_1 est dite représentative pour c_j au point x si et seulement si

$$\beta(c_j(x,\phi)) = \beta(c_j(x,1)) \quad \forall \phi \in \Phi \text{ tel que } \phi \geq \phi_1.$$

Autrement dit, pour un certain x et un certain c_j, les fidélités représentatives sont celles supérieures à la plus grande fidélité n'identifiant pas correctement si la contrainte est satisfaite

ou non. L'analyse consiste à déterminer les estimations suivantes.

$$\begin{aligned} r_{ij} :=&\ \text{Estimation de la probabilité que la fidélité } \phi_i \in \Phi \\ &\ \text{soit représentative pour la contrainte } c_j. \\ p_{ij} :=&\ \text{Estimation de la probabilité que } c_j \text{ soit satisfaite à } \phi_i \in \Phi \\ =&\ Pr[\beta(c_j(x, \phi_i)) = 0] \text{ pour un } x \in X \text{ quelconque.} \\ t_i :=&\ \text{Estimation du temps d'une évaluation à } \phi_i \in \Phi. \end{aligned}$$

Soit H, l'ensemble contenant tous les points d'un hypercube latin. Ces estimations sont obtenues en calculant des proportions ou des moyennes sur les points de H. Plus l'échantillon est grand, plus ces proportions et moyennes sont près des vraies probabilités et des vraies moyennes respectivement.

$$\begin{aligned} r_{ij} &= \frac{1}{|\mathrm{H}|}|\{x \in \mathrm{H} : \phi_i \text{ est représentative pour } c_j\}| && \forall i \in I\,, \forall j \in J \\ p_{ij} &= \frac{1}{|\mathrm{H}|}|\{x \in \mathrm{H} : \beta(c_j(x, \phi_i)) = 0\}| && \forall i \in I\,, \forall j \in J \\ t_i &= \frac{1}{|\mathrm{H}|} \sum_{x \in \mathrm{H}} t(x, \phi_i) && \forall i \in I. \end{aligned}$$

4.2 Modèle d'optimisation de la matrice de biadjacence

Étant données les estimations précédentes, un modèle d'optimisation est proposé pour déterminer la matrice de biadjacence qui minimise l'espérance du temps requis par un lancement de l'algorithme 3.2. Dans ce modèle, un dernier paramètre $\varepsilon \in [0, 1]$ que l'utilisateur doit choisir apparaît. Il s'agit de la borne supérieure posée sur la probabilité estimée qu'une contrainte c_j avec $j \in J$ soit mal représentée. À chaque lancement de l'algorithme 3.2, des sous-évaluations sont exécutées. Celles-ci correspondent aux variables y_i dans le modèle. Pour chaque fidélité $\phi_i \in \Phi$, $y_i = 1$ si une sous-évaluation est effectuée, et $y_i = 0$ sinon. Durant une sous-évaluation, les c_j qui sont vérifiées sont données par B.

$$
(P)\qquad
\begin{aligned}
\min_{B\in\mathbb{B}^{\ell\times m},y\in\mathbb{R}^{\ell}} \quad & t_1y_1+\sum_{i=2}^{\ell}\left(t_iy_i\prod_{k=1}^{i-1}\prod_{j\in J}p_{kj}B_{kj}+1-B_{kj}\right) && (4.1)\\
\text{s.c.} \quad & \sum_{i\in I}B_{ij}=1 \qquad \forall\, j\in J && (4.2)\\
& r_{ij}\geq B_{ij}-\varepsilon \quad \forall\, i\in I,\,\forall\, j\in J && (4.3)\\
& B_{ij}\leq y_i\leq 1 \quad \forall\, i\in I,\,\forall\, j\in J && (4.4)\\
& y_i\leq \sum_{j\in J}B_{ij} \qquad \forall\, i\in I. && (4.5)
\end{aligned}
$$

Le problème (P) est un problème de minimisation. Toutefois, simplement minimiser le temps se résume à assigner toutes les contraintes à la plus petite fidélité. Intuitivement, le modèle devrait minimiser le temps et maximiser la probabilité que les contraintes donnent la même valeur de la fonction $\beta(\cdot)$ à leur fidélité assignée et à la vérité. Or, les modèles d'optimisation bi-objectifs sont plus complexes. Il est préférable d'uniquement minimiser l'espérance du temps, et d'inclure la contrainte (4.3) qui pose une borne supérieure ε sur la probabilité estimée qu'une c_j soit mal représentée.

La fonction objectif (4.1) est une somme de temps, chacun multiplié par la probabilité que ce temps soit ajouté à l'évaluation. La première sous-évaluation est effectuée avec une probabilité de y_1, qui vaut 1 ou 0 : on a alors t_1y_1. Le temps t_i avec $i\geq 2$ est ensuite ajouté à l'objectif si $y_i=1$, donc si une sous-évaluation est effectuée à ϕ_i, et si les sous-évaluations aux fidélités inférieures n'ont pas causé d'interruption. La fonction objectif est alors donné par

$$
\begin{aligned}
f =\, & t_1y_1\\
& + t_2y_2Pr[\text{pas d'interruption à } \phi_1]\\
& + t_3y_3Pr[\text{pas d'interruption à } \phi_1]Pr[\text{pas d'interruption à } \phi_2]\\
& + t_4y_4Pr[\text{pas d'interruption à } \phi_1]Pr[\text{pas d'interruption à } \phi_2]Pr[\text{pas d'interruption à } \phi_3]\\
& + \dots\\
=\, & t_1y_1+\sum_{i=2}^{\ell}\left(t_iy_i\prod_{k=1}^{i-1}Pr[\text{pas d'interruption à } \phi_k]\right). \qquad (4.6)
\end{aligned}
$$

Les indices k dénotent des indices inférieurs à un certain $i\in I$. La probabilité qu'il n'y ait pas d'interruption à ϕ_k correspond à la probabilité que chaque contrainte assignée à cette fidélité soit satisfaite. Cette probabilité correspond à la multiplication des probabilités que

chaque contrainte soit satisfaite individuellement, qui sont estimées par p_{kj}. La formulation

$$Pr[\text{pas d'interruption à } \phi_k] = \prod_{j \in J} p_{kj} B_{kj} + 1 - B_{kj} \tag{4.7}$$

est telle que si $B_{ij} = 0$, le j-ième élément de la multiplication est égal à 1, et si $B_{ij} = 1$, signifiant que la contrainte sera vérifiée par l'algorithme, le j-ième élément vaut p_{ij}. En substituant (4.7) dans (4.6), on obtient la forme finale (4.1). Pour donner un exemple, la solution de la figure 3.2 est telle que

$$\begin{aligned}
f =& t_1 + 0 + t_3 \prod_{k=1}^{2} Pr[\text{pas d'interruption à } \phi_k] + t_4 \prod_{k=1}^{3} Pr[\text{pas d'interruption à } \phi_k] \\
=& t_1 + t_3 \left(\prod_{j \in J} p_{1j} B_{1j} + 1 - B_{1j} \right) \left(\prod_{j \in J} p_{2j} B_{2j} + 1 - B_{2j} \right) \\
& \qquad + t_4 \left(\prod_{j \in J} p_{1j} B_{1j} + 1 - B_{1j} \right) \left(\prod_{j \in J} p_{2j} B_{2j} + 1 - B_{2j} \right) \left(\prod_{j \in J} p_{3j} B_{3j} + 1 - B_{3j} \right) \\
=& t_1 + t_3 (p_{11} p_{12})(1) + t_4 (p_{11} p_{12})(1)(p_{33} p_{35}) \\
=& t_1 + t_3\, p_{11} p_{12} + t_4\, p_{11} p_{12} p_{33} p_{35}.
\end{aligned}$$

L'équation 4.2 assure que chaque contrainte est assignée à une seule fidélité. Autrement dit, B doit être telle que le degré de chaque sommet c_j dans G est égal à 1. L'équation 4.3 assure qu'aucune contrainte ne peut être assignée à une fidélité où la probabilité estimée que la contrainte soit mal représentée excède ε. Si c_j n'est pas assignée à ϕ_i, $B_{ij} = 0$ et la contrainte est nécessairement satisfaite. Un élément B_{ij} ne peut être égal à 1 que si la représentativité r_{ij} est supérieure ou égale à $1 - \varepsilon$. Par exemple, avec $\varepsilon = 0.05$, c_j ne peut être assignée qu'aux ϕ_i telles que $r_{ij} \geq 0.95$. Le choix de ε offre un compromis. Plus cette valeur et grande, plus les sous-évaluations sont effectuées à basse fidélité, mais plus il y a de chances que des interruptions surviennent sur des points réalisables. Inversement, un petit ε assure des interruptions plus justifiées, au prix de sous-évaluations plus longues. Finalement, les équations (4.4) et (4.5) assurent que $y_i = 1$ si au moins une contrainte c_j est assignée à ϕ_i, et que $y_i = 0$ sinon.

4.3 Différences entre le modèle et le vrai problème

Le problème (P) ne modélise pas exactement tous les éléments de l'algorithme 3.2. Les omissions au modèle sont listées et justifiées.

— L'ajout d'une sous-évaluation à $\phi = 1$ si la dernière sous-évaluation n'est pas effectuée à $\phi = 1$ et si le point en évaluation est le meilleur depuis le début de l'optimisation augmente le temps espéré. Or, il est impossible de prévoir à quels moments cette sous-évaluation survient, ce temps est donc impossible à ajouter au modèle.

— Si des contraintes à priori font partie du problème et ne sont pas considérées dans le modèle, certaines sous-évaluations auront un temps approximativement nul, ce qui diminue l'espérance du temps. En revanche, cette différence n'a aucun impact sur la solution optimale.

— Tel que le modèle est présenté, il est assumé que pour un point quelconque, si c_j est satisfaite à une ϕ_i où $B_{ij} = 1$, elle est satisfaite à toute fidélité supérieure à ϕ_i. Cela n'est pas toujours vrai.

Autrement dit, le modèle exprime un algorithme qui ne vérifie que les c_j assignées à ϕ_i à chaque sous-évaluation. Toutefois, l'algorithme 3.2 vérifie également les c_j assignées à des fidélités inférieures. Par exemple, dans le problème de la figure 3.2, sans considérer c_3 et c_5, la sous-évaluation à ϕ_4 a lieu si c_1 et c_2 sont satisfaites à ϕ_1 (probabilité estimée de $p_{11}p_{12}$) et si elles sont encore satisfaites à ϕ_3 (probabilité estimée de $p_{31}p_{32}$). Or, pour multiplier ces probabilités, il faut considérer la probabilité p_{31} sachant que $\beta(c_1(x, \phi_1)) = 1$, et la probabilité p_{32} sachant que $\beta(c_2(x, \phi_1)) = 1$ pour un x quelconque. L'inconvénient est que cela complexifie grandement le modèle.

Deux hypothèse peuvent être posées pour éviter ce problème. La première suppose que les probabilités p_{ij} sont toutes indépendantes. Avec cette hypothèse, dans la fonction objectif (4.1), à chaque t_i où sont multipliés les p_{ij} où $B_{ij} = 1$, il faut également multiplier les probabilités associées aux contraintes assignées à des fidélités ϕ_q inférieures à ϕ_i qui sont revérifiées par l'algorithme à chaque sous-évaluation. La fonction objectif est alors donnée par

$$f = t_1 y_1 + \sum_{i=2}^{\ell} \left(t_i y_i \prod_{q=1:y_q=1}^{i-1} \prod_{k=1}^{q} \prod_{j \in J} p_{qj} B_{kj} + 1 - B_{kj} \right).$$

Dans l'exemple de la figure 3.2, la fonction objectif devient

$$f = t_1 + t_3\, p_{11}p_{12} + t_4\, p_{11}p_{12}p_{31}p_{32}p_{35}p_{33}.$$

La seconde hypothèse stipule que si une contrainte est satisfaite à une fidélité à laquelle elle est assignée, elle le sera encore pour toute fidélité supérieure. C'est l'hypothèse qui est posée pour déterminer la fonction objectif (4.1) du problème (P). Il est à noter que plus ε est proche de zéro, plus cette hypothèse est proche de la réalité (à $\varepsilon = 0$ l'hypothèse est valide). Pour un certain point lors d'une évaluation, parmi les fidélités qui ont une haute représentativité,

il est rare (au pire, dans $\varepsilon\%$ des cas) qu'une contrainte ne donne pas la même valeur de $\beta(\cdot)$ que la vérité. C'est donc l'hypothèse qui est favorable, à moins qu'un grand ε soit choisi.

4.4 Résolution rapide du modèle

Le modèle d'optimisation (P) comporte une fonction objectif polynomiale et des variables binaires. Il s'agit d'un problème qui peut être long à résoudre et sans garantit sur l'optimalité pour les solveur existants. Cette section propose une méthode adaptée au problème de biadjacence optimale qui permet de trouver une solution rapidement. Puisque les B_{ij} sont des variables binaires et que chaque y_i est égal à 0 ou 1 pour les solutions réalisables, le nombre de solutions est fini. Quatre façons de réduire la taille de l'espace des solutions sans exclure toutes les solutions optimales (à part dans un cas rare discuté plus bas) sont montrées. Ces méthodes réduisent l'espace des solutions à un point tel qu'une recherche exhaustive suffit pour résoudre le problème en trouvant la meilleure solution.

La première réduction consiste à omettre les variables y_i ainsi que les équations (4.4) et (4.5). Comme on vise à évaluer la valeur de la fonction objectif de chaque solution restante après les réductions, les y_i peuvent être simplement calculés à partir d'une solution B comme suit :

$$y_i = \begin{cases} 0 & \text{si } \sum\limits_{j \in J} B_{ij} = 0 \\ 1 & \text{sinon} \end{cases} \qquad \forall i \in I. \tag{4.8}$$

La seconde réduction repose sur l'idée que si une contrainte est à priori, il est inutile de l'inclure dans le modèle. En effet, les contraintes à priori n'ont aucun impact sur la matrice de biadjacence optimale. Il est alors possible de diminuer le nombre de sommets dans G en remplaçant J par

$$\hat{J} := \{j \in J : c_j \text{ n'est pas une contrainte à priori}\}. \tag{4.9}$$

La troisième réduction consiste également à diminuer le nombre de sommets dans G, et donc à diminuer la taille de B. L'ensemble I est remplacé par

$$\hat{I} := \bigcup_{j \in J} \min\{i \in I : r_{ij} \geq 1 - \varepsilon\}. \tag{4.10}$$

Le théorème 3 indique que si une solution réalisable existe, une solution optimale qui est telle que toutes les contraintes sont assignées à des $\phi_i : i \in \hat{I}$ existe. Il est alors possible

de ne pas considérer les $\phi_i \in I \backslash \hat{I}$ dans le modèle d'optimisation sans exclure toutes les solutions optimales. Cette réduction est généralement celle qui permet de diminuer le plus significativement l'espace des solutions. Comme ce résultat est important, le théorème 3 est démontré. Le théorème s'applique sous l'hypothèse que

$$p_{aj} \leq p_{bj} \quad \forall a, b \in I : a < b \text{ et } \exists B \text{ réalisable où } B_{aj} = 1, \forall j \in J. \tag{4.11}$$

Cette hypothèse est discutée après la démonstration. Deux propriétés sont exploitées par ce théorème. La première est qu'étant donné une contrainte c_j assignée à une fidélité ϕ_i où $r_{ij} \geq B_{ij} - \varepsilon$ (la contrainte 4.3) est respectée, la contrainte est également respectée pour toute fidélité supérieure à ϕ_i. Ce résultat est vrai par la définition de la représentativité :

$$r_{aj} \geq 1 - \varepsilon \implies r_{bj} \geq 1 - \varepsilon \quad \forall j \in J, \forall \text{ paire } a, b \in I : b \geq a. \tag{4.12}$$

La seconde propriété est que le temps est une fonction monotone croissante selon la fidélité :

$$\phi_a < \phi_b \implies t_a \leq t_b \quad \forall \text{ paire } a, b \in I. \tag{4.13}$$

Cette propriété provient de la seconde définition du tableau 3.1. Sans cette propriété, il peut exister une fidélité $\phi_i \in \Phi$ supérieure à la plus petite fidélité dans Φ pour laquelle le temps moyen d'exécution de la boîte noire est le plus bas, et le théorème ne tient plus si $i \notin \hat{I}$.

Dans les lemmes et le théorème qui suivent, seulement les variables B_{ij} sont prises en compte puisque les variables y_i peuvent toujours être calculées à partir de B avec (4.8), et ce, de façon à ce que (4.4) et (4.5) soient toujours satisfaites.

Lemme 1

Soit B une solution réalisable pour (P) où il existe un $i' \in I \backslash \hat{I}$ tel qu'au moins une contrainte c_j est assignée à $\phi_{i'}$, et B' une solution identique à B à l'exception que toute contrainte assignée à $\phi_{i'}$ est réassignée à $\phi_{i'-1}$. Formellement,

$$B'_{ij} = \begin{cases} 0 & \text{si } i = i' \\ 1 & \text{si } i = i' - 1 \text{ et } B_{i'j} = 1 \\ B_{ij} & \text{sinon.} \end{cases} \tag{4.14}$$

La solution B' est réalisable pour (P).

Démonstration

La solution B' respecte (4.3) :

$$B \text{ est réalisable } \implies r_{i'j} \geq 1-\varepsilon \quad \forall j \in J \text{ où } B_{i'j} = 1.$$

Pour toute contrainte c_j, il existe une fidélité minimale pour laquelle $r_{ij} \geq 1-\varepsilon$. L'ensemble $\hat{I}$ contient ces fidélités minimales. Sachant que $i' \notin \hat{I}$,

$$\begin{aligned} & i' > \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J \text{ où } B_{i'j} = 1 \\ \implies & i'-1 \geq \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J \text{ où } B_{i'j} = 1. \end{aligned}$$

L'équation (4.12) indique que lorsque $r_{ij} \geq 1-\varepsilon$ pour une fidélité ϕ_i et une contrainte c_j, elle l'est pour toute fidélité supérieure à ϕ_i pour c_j. Il s'ensuit que

$$\begin{aligned} & r_{i'-1\,j} \geq 1-\varepsilon && \forall j \in J \text{ où } B_{i'j} = 1 \\ \implies & r_{i'-1\,j} \geq 1-\varepsilon && \forall j \in J \text{ où } B'_{i'-1\,j} = 1 \\ \implies & (4.3) \text{ est satisfaite} && \forall j \in J \text{ où } B'_{i'-1\,j} = 1. \end{aligned}$$

Pour toute autre contrainte c_j, comme $B'_{ij} = B_{ij}$ et B est réalisable, B' respecte (4.3). La solution B' respecte (4.2) :

$$\begin{aligned} B^b \text{ est réalisable } \implies 1 &= \sum_{i \in I} B_{ij} && \forall\, j \in J \\ &= \sum_{i=1}^{i'-2} B_{ij} + 0 + 1 + \sum_{i=i'+1}^{l} B_{ij} && \forall j \in J : \text{ où } B_{i'j} = 1 \\ &= \sum_{i=1}^{i'-2} B'_{ij} + 1 + 0 + \sum_{i=i'+1}^{l} B'_{ij} && \forall j \in J \text{ où } B'_{i'-1\,j} = 1 \\ &= \sum_{i \in I} B'_{ij} && \forall\, j \in J \\ \implies \quad & B' \text{ respecte (4.2)} \end{aligned}$$

La solution B^a respecte (4.3) et (4.2), ce qui implique que B' est réalisable pour (P).

□

Lemme 2

Soit B une solution réalisable pour (P) où il existe un $i' \in I \backslash \hat{I}$ tel qu'au moins une contrainte c_j est assignée à $\phi_{i'}$, et B^a une solution définie par (4.14). L'équation $f(B^a) \leq f(B^b)$ est vraie.

Démonstration

Adoptons d'abord les définitions suivantes qui simplifieront la notation. Étant donné une solution B^s, les y_i^s sont les y_i donnés par cette solution et

$$M_i^s := y_i^s \prod_{k=1}^{i-1} \prod_{j \in J} p_{ij}\, B_{kj}^s + 1 - B_{kj}^s. \tag{4.15}$$

Dans les développements qui suivent, la fonction objectif est séparée en une sommation de cinq termes. À chaque étape, un ou plusieurs termes est modifié. Deux scénarios peuvent survenir, dépendamment de la valeur de $y_{i'-1}^b$. Si $y_{i'-1}^b = 0$:

$$\begin{aligned}
f(B^b) =& t_1 y_1^b + \sum_{i=2}^{i'-2} t_i M_i^b + t_{i'-1} M_{i'-1}^b + t_{i'} M_{i'}^b + \sum_{i=i'+1}^{l} t_i M_i^b \\
=& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a + t_{i'-1} M_{i'-1}^b + t_{i'} M_{i'}^b + \sum_{i=i'+1}^{l} t_i M_i^b \\
& \qquad \text{car } B_{ij}^b = B_{ij}^a \, \forall i \in I \leq i'-2, \, \forall j \in J \\
\geq& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a + t_{i'-1} M_{i'-1}^b + t_{i'} M_{i'}^b + \sum_{i=i'+1}^{l} t_i M_i^a \qquad (4.16) \\
& \qquad \text{car (4.11) et (4.13)} \\
=& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a + t_{i'-1} M_{i'-1}^b + t_{i'} M_{i'-1}^a + \sum_{i=i'+1}^{l} t_i M_i^a \\
& \qquad \text{car } M_{i'}^b = M_{i'-1}^a \\
\geq& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a + t_{i'-1} M_{i'-1}^b + t_{i'-1} M_{i'-1}^a + \sum_{i=i'+1}^{l} t_i M_i^a \\
& \qquad \text{car } t_{i'} \geq t_{i'-1} \text{ car (4.13)} \\
=& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a + t_{i'-1} M_{i'-1}^a + t_{i'} M_{i'}^a + \sum_{i=i'+1}^{l} t_i M_i^a \\
& \qquad \text{car } t_{i'} M_{i'}^a = t_{i'-1} M_{i'-1}^b = 0 \\
=& f(B^a) \qquad (4.17)
\end{aligned}$$

Si $y^b_{i'-1} = 1$:

$$f(B^b) \geq t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^b_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i M^a_i \tag{4.18}$$

$$\text{car (4.16)}$$

$$= t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^a_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i M^a_i$$

$$\text{car } t_{i'-1} M^b_{i'-1} = t_{i'-1} M^a_{i'-1}$$

$$> t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'} M^a_{i'} + t_{i'-1} M^a_{i'-1} + \sum_{i=i'+1}^{l} t_i M^a_i$$

$$\text{car } t_{i'} M^b_{i'} > 0 = t_{i'} M^a_{i'}$$

$$= f(B^a) \tag{4.19}$$

$$(4.17) \text{ et } (4.19) \implies f(B^a) \leq f(B^b)$$

□

Théorème 3

Dans l'argmin d'un problème de matrice de biadjacence optimale, à moins que $\Omega = \varnothing$, il existe toujours une solution où toutes les contraintes sont assignées à des fidélité $\phi_i : i \in \hat{I}$.

Démonstration

Si Ω n'est pas vide, l'argmin peut ne contient que des solutions où toutes les contraintes sont assignées à des fidélité $\phi_i : i \in \hat{I}$. Le théorème est alors trivial. Si ce n'est pas la cas,

$$\exists B^* \in \underset{B}{\operatorname{argmin}}\{f(B) : B \in \Omega\} : \exists (i \notin \hat{I}, j \in J) \text{ où } B^*_{ij} = 1.$$

Le lemme 2 montre qu'il existe une solution différente de B^* qui donne un objectif de valeur égale ou meilleure. Le lemme 1 montre que cette autre solution est réalisable. Cela implique qu'elle appartient également à l'argmin. Cette nouvelle solution est obtenue en réassignant les contraintes assignées à un $\phi_{i'} : i' \notin \hat{I}$ à $\phi_{i'-1}$. Par induction, plusieurs réassignations peuvent être effectuées consécutivement jusqu'à ce que toutes les contraintes soient assignées à des fidélité $\phi_i : i \in \hat{I}$. Dès lors,

$$\exists B^* \in \underset{B}{\operatorname{argmin}}\{f(B) : B \in \Omega\} : B^*_{ij} = 0 \, \forall i \notin \hat{I}, \, \forall j \in J.$$

□

L'hypothèse (4.11) assure que l'inégalité (4.16) du lemme 2 est vraie. Cette inégalité est aussi utilisée par (4.18). Considérons l'exemple de problème suivant où l'hypothèse est fausse, avec r la matrice des r_{ij} et p la matrice des p_{ij}. Dans cet exemple, $\varepsilon = 0.05$, $\ell = 3$, $m = 2$, B^b est une solution telle que c_1 est assignée à $\phi_{i'}$ où $i' = 2$ et B^a est une solution identique à B^b à l'exception que c_1 est assignée à $\phi_{i'-1}$.

$$r = \begin{bmatrix} 0.95 & 0.7 \\ 1 & 0.7 \\ 1 & 1 \end{bmatrix} \quad p = \begin{bmatrix} 0.53 & 0.4 \\ 0.5 & 0.4 \\ 0.5 & 0.4 \end{bmatrix} \quad B^b = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B^a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.20}$$

L'hypothèse est fausse puisque $p_{11} > p_{21}$, et B^b et B^a sont toutes deux réalisables. En reprennant la notation du lemme 2, les objectifs des solutions sont les suivants.

$$\begin{aligned} f(B^b) = & \underbrace{0}_{t_1 y_1^b + \sum_{i=2}^{i'-2} t_i M_i^b} + \underbrace{0}_{t_{i'-1} M_{i'-1}^b} + \underbrace{t_2}_{t_{i'} M_{i'}^b} + \underbrace{t_3\, p_{21}}_{\sum_{i=i'+1}^{\ell} t_i M_i^b} \\ f(B^a) = & \underbrace{0}_{t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a} + \underbrace{t_1}_{t_{i'-1} M_{i'-1}^a} + \underbrace{0}_{t_{i'} M_{i'}^a} + \underbrace{t_3\, p_{11}}_{\sum_{i=i'+1}^{\ell} t_i M_i^a} \end{aligned}$$

On à alors que

$$\begin{aligned} f(B^b) &= t_2 + t_3\, p_{21} \\ &< t_2 + t_3\, p_{11} = t_1 y_1^a + \sum_{i=2}^{i'-2} t_i M_i^a + t_{i'-1} M_{i'-1}^b + t_{i'} M_{i'}^b + \sum_{i=i'+1}^{l} t_i M_i^a, \end{aligned}$$

ce qui contredit (4.16). Cela dit, il est possible que le résultat $f(B^b) \geq f(B^a)$ du lemme 2 soit tout de même conservé.

$$\begin{aligned} f(B^b) \geq f(B^a) &\iff t_2 + t_3\, p_{21} \geq t_1 + t_3\, p_{11} \\ &\iff t_2 - t_1 \geq (p_{11} - p_{21})\, t_3 \end{aligned} \tag{4.21}$$

Si l'hypothèse était satisfaite ($p_{11} \leq p_{21}$), l'inégalité $f(B^b) \geq f(B^a)$ serait trivialement satisfaite. Dans l'exemple donné, l'inégalité est vraie si le temps gagné par la réassignation de c_1 est plus significatif que la perte de probabilité d'effectuer une interruption qui coupe t_3.

Dès lors, on peut s'intéresser à déterminer des bornes supérieure et inférieure sur cette perte de probabilité. Observons d'abord que les valeurs des p_{ij} et des r_{ij} sont reliées, en supposant que p_{21} et p_{11} ne sont pas donnés dans l'exemple. Dans ce dernier, $r_{\ell 1} = 1$ et $p_{\ell 1} = 0.5$. Cela signifie qu'à la fidélité correspondant à la vérité, la moitié des points de l'échantillon respectent c_1. Avec $r_{21} = 1$, tous les points identifient correctement si c_1 est satisfaite ou non en vérité à la fidélité ϕ_2. Sachant que cette proportion est de 0.5, p_{21} vaut certainement 0.5. Avec $r_{11} = 0.95$, seulement 5% des points de l'échantillon n'identifient pas correctement si c_1 est satisfaite en vérité à la fidélité ϕ_1. Le plus grand p_{11} possible est observé si ce 5% des points ne satisfont pas c_1 en vérité, et satisfont c_1 à ϕ_1. Dans ce cas, $p_{11} = 0.55$. Inversement, le plus petit p_{11} possible est observé si le 5% des points satisfont c_1 en vérité, et ne satisfont pas c_1 à ϕ_1. Dans ce cas, $p_{11} = 0.45$. Autrement dit, pour toute contrainte c_j, la différence $r_{\ell j} - r_{ij}$ est une mesure de la différence maximale entre p_{ij} et $p_{\ell j}$. Pour déterminer les bornes supérieure et inférieure, on cherche à trouver la plus grande différence $r_{\ell j} - r_{ij}$ dans le cas général (aucun r_{ij} ni p_{ij} donné). Pour toute contrainte c_j, $r_{\ell j} = 1$. Comme l'hypothèse ne concerne que les solutions réalisables, on s'intéresse aux $(i, j) \in (I, J)$ où $r_{ij} \geq 1 - \varepsilon$. Cela implique que $r_{\ell j} - r_{ij} \leq \varepsilon$. Ce raisonnement comporte toutefois quelques limites. Si la différence maximale entre un p_{ij} et un $p_{\ell j}$ est supérieure à $p_{\ell j}$ ou supérieure à $1 - p_{\ell j}$, d'autres phénomènes entres en jeux, sans quoi il est possible que $p_{ij} > 1$ ou que $p_{ij} < 0$, ce qui est absurde. Formellement,

$$\varepsilon \leq p_{\ell j} \text{ et } \varepsilon \leq 1 - p_{\ell j} \implies p_{ij} \in [p_{\ell j} - \varepsilon, p_{\ell j} + \varepsilon] \quad \forall (i, j) \in (I, J) : r_{ij} \geq 1 - \varepsilon.$$

Pour revenir à l'exemple, si p_{11} n'était pas donné, il est tout de même connu que $p_{11} \in [0.45, 0.55]$. La perte de probabilité $p_{11} - p_{21}$ d'effectuer une interruption à ϕ_3 est alors d'au plus $\varepsilon = 0.05$. Pour résumer, l'hypothèse (4.11) assure que le théorème 3 implique que la réduction proposée par (4.10) n'exclue pas de solution optimale. Aussi, même si l'hypothèse est fausse, la réduction n'exclue pas de solution optimale si les réassignations du théorème 3 sont telles que le gain en temps est plus significatif que la perte de probabilité d'effectuer des interruptions. Finalement, plus ε est petit, moins cette perte de probabilité est significative.

La quatrième réduction consiste à retirer les solutions non réalisables. Avec les réductions (4.10) et (4.9), on obtient un graphe $\hat{G}$ avec une quantité diminuée de sommets. La contrainte (4.3) indique que certaines arêtes ne peuvent pas exister. Définissons la matrice $\hat{B}$ comme la matrice de biadjacence de $\hat{G}$ telle que les $\hat{B}_{ij} : r_{ij} < 1 - \varepsilon$ sont fixés à 0. Finalement, (4.2) indique que les sommets de $\{c_1, c_2, \ldots, c_m\}$ ont un degré égal à 1. Il est alors possible de lister exhaustivement toutes les solutions $\hat{B}$ possibles qui respectent la contrainte sur les degrés. Pour chaque solution dans cette liste, l'objectif (4.1) est calculé, et les solutions qui donnent la plus petite valeur de l'objectif sont optimales. Une fois cette solution trouvée, il

est possible de reconstruire B en ajoutant les colonnes $J \backslash \hat{J}$ et les rangées $I \backslash \hat{I}$ à $\hat{B}$, en donnant à chacun de ces nouveaux éléments une valeur nulle. On obtient ainsi B^*, une solution optimale de (P). Cette matrice de biadjacence permet de déterminer E, et la dernière étape de l'algorithme 3.1 peut être lancée.

4.5 Algorithme d'optimisation avec contraintes hiérarchisées détaillé

Maintenant que chaque étape de l'algorithme 3.1 a été détaillée, l'algorithme 4.1 présente une version plus complète de celui-ci. L'objet BB dénote la boîte noire à optimiser qui contient n variables, l'espace des variables X, une fonction objectif f, une fonction *Sous-évaluation* pour l'évaluer, et m contraintes. La fonction *Évaluation* provient de l'algorithme 3.2.

L'algorithme 4.1 peut être adapté au cas où l'utilisateur ne connait pas de bon point de départ. D'abord, les bornes de l'hypercube latin sont définies par X, peu importe la valeur de Δ. Après avoir effectué les évaluations, il suffit de définir le meilleur point de l'échantillon comme étant x^0, puis de considérer uniquement les points de l'hypercube latin qui sont à l'intérieur des bornes inférieures l^{cen} et des bornes supérieures u^{cen} pour le calcul des r_{ij}, des p_{ij} et des t_i. Toutefois, cette méthode ne fonctionne que pour les problèmes de petite dimension ou avec un grand Δ. Autrement, il est possible que le nombre de points contenus par l^{cen} et u^{cen} soit trop petit. Si, par exemple, un problème est de dimension $n = 3$ et que l'utilisateur choisit $\Delta = 1/2$, le volume de l'échantillon considéré pour l'analyse du comportement des contraintes est le volume de l'hypercube divisé par deux dans trois axes. De façon générale, le volume de l'hypercube est divisé par au moins Δ^{-n}. Le vomule est réduit davantage si au moins une borne $u_i^{\text{cen}} > u_i$ ou si au moins une borne $l_i^{\text{cen}} < l_i$.

Attention, est-ce que B est une solution de (P) ou est-ce que c'est plutôt (B, y) (À vérifier pour le chapitre 4 au complet quand la notation sera finale).

Algorithme 4.1 : Optimisation avec contraintes hiérarchisées (détaillé)

Fonction *Optimisation*$(x^0, \mathrm{BB}, \Delta, \Phi, \varepsilon, |H|)$

Si $1 \notin \Phi$

> **Renvoyer** un message d'erreur spécifiant qu'il est requis que $1 \in \Phi$.

1. Analyse du comportement des contraintes en fonction de la fidélité.

> Déterminer l^{cen} et u^{cen} à partir de X, Δ et x^0.
>
> Déterminer les points de H, l'hypercube latin de $|H|$ points borné par l^{cen} et u^{cen}.
>
> Évaluer les points de H à chaque fidélité de Φ, en parallélisant les évaluations autant que possible.
>
> Calculer les r_{ij}, les p_{ij}, et les t_i.

2. Calcul d'une matrice de biadjacence optimale.

> Construire le problème (P).
>
> Calculer B^*, une solution optimale de (P) avec la méthode proposée à la section 4.4 avec ε.
>
> Construire le graphe biparti $G = (V_1, V_2, E)$ où $V_1 = \Phi$, $V_2 = \{c_1, c_2, \ldots, c_m\}$ et E est donné par B^*.

3. Optimisation de la boîte noire.

> Lancer l'optimisation avec un solveur existant, en fournissant comme problème au solveur la fonction $\acute{E}valuation(x, Sous\text{-}évaluation(\cdot), G)$ ainsi que x^0.

Renvoyer les sorties du solveur.

CHAPITRE 5 RÉSULTATS

Le cinquième chapitre présente les résultats numériques de quelques implémentations de la méthode d'optimisation avec contraintes hiérarchisées. Comme la boîte noire de PRIAD n'est pas terminée au moment de l'écriture de ce mémoire, les tests sont effecués avec la famille de boîtes noires SOLAR. Ces boîtes noires sont décrites par Lemyre Garneau (2015), et elles ont été développées pour offrir un banc d'essais aux algorithmes d'optimisation de boîtes noires. La famille de boîtes noires SOLAR simule une centrale de production électrique solaire thermodynamique. Aussi, il est décrit comment la méthode pourrait potentiellement être implémentée avec le problème de PRIAD. Finalement, le chapitre se termine avec une discussion des résultats présentés.

5.1 Banc d'essais et implémentations

En particulier, quatre problèmes comportent un aspect multi-fidélité : solar2, solar3, solar4 et solar7. Ces différentes boîtes noires constituent des simulations de différentes parties de la centrale solaire thermodynamique. Leur différentes charactéristiques sont montrées au tableau 5.1. La colonne m_{ap} dénote le nombre de contraintes à priori, $m_{\text{multi-}\phi}$ dénote le nombre de contraintes affectées par la multi-fidélité, f_{an} indique si l'objectif est formulé analytiquement et $f_{\text{multi-}\phi}$ indique si l'objectif est affecté par la multi-fidélité. Une contrainte affectée par la multi-fidélité est nécessairement simulée, et n'est donc pas à priori.

Boîte noire	n	m	m_{ap}	$m_{\text{multi-}\ phi}$	f_{an}	$f_{\text{multi-}\phi}$
solar2	14	13	5	4	oui	non
solar3	20	13	5	5	non	non
solar4	29	16	7	6	non	non
solar7	7	6	2	2	non	oui

TABLEAU 5.1 Charactéristiques des quatre boîtes noires SOLAR multi-fidélité.

Le problème solar7 contient peu de contraintes, et celles-ci sont faciles à satisfaire. Il s'agit d'un problème qui permet d'observer comment la méthode se comporte lorsqu'elle n'a pas

beaucoup de moyens permettant d'effectuer des interruptions. De plus, n'ayant que 7 variables, solar7 est un problème que les solveurs existants peuvent résoudre efficacement. Les problèmes solar2 et solar3 comportent des contraintes difficiles à satisfaires. Il s'agit de boîtes noires où la méthode testée a beaucoup de potentiel. En ce qui concerne solar4, les contraintes sont significativement plus difficiles a satisfaire. Cette boîte noire permet de tester la méthode dans un environnement où simplement trouver un point réalisable est d'une grande difficulté. Aussi, comportant 29 variables, solar4 est un problème qui est significativement plus difficile pour les solveurs existants. Ces propos sont supportés par le tableau 5.2 qui montre la quantité de points qui satisfont les contraintes à priori, et la quantité de points qui satisfont toutes les contraintes d'un hypercube latin de 10^4 points. Ces points sont évalués à $\phi = 1$ seulement.

Boîte noire	**Réalisables à priori**	**Réalisables**
solar2	1853	1
solar3	894	1
solar4	10	0
solar7	4608	1354

TABLEAU 5.2 Réalisabilité de 10^4 points d'un hypercube latin pour 4 instances SOLAR.

Un seul point est réalisable pour les problèmes solar2 et solar3, ce qui indique qu'elles comportent en effet des contraintes difficiles à satisfaire. Le problème solar4 est tel que les solutions réalisables sont particulièrement rares. Seulement 10 points sur 10^4 satisfont les 7 contraintes à priori, et aucun ne satisfait les 16 contraintes. Finalement, solar7 comprend une grande proportion de solutions réalisables.

Une remarque importante est que parmi les fonctions objectif des quatre boîtes noires, seulement celle de solar7 est affectée par la multi-fidélité. Conséquemment il est imposé pour solar7 que la dernière sous-évaluation effectuée par la passerelle soit avec une fidélité de 1. Autrement, il est possible que l'information donnée au solveur sur f pour les points réalisables soit érronée. En imposant une sous-évaluation à une fidélité de 1, il est garanti que la valeur de $f(x)$ donnée au solveur pour tout x réalisable soit la vérité. Au niveau de l'implémentation, cela signifie qu'il doit être imposé que $1 \in \Phi$ et $y_\ell = 1$ lors du calcul de la matrice de biadjacence optimale. Cela indique dans le modèle que cette sous-évaluation a toujours lieu,

et on cherche à touver la matrice de biadjacence optimale sachant que la sous-évaluation à $\phi = 1$ aura lieu.

La méthode d'optimisation avec contraintes hiérarchisées doit être couplée à un solveur existant. Tous les tests sont effectués avec le solveur NOMAD, qui est présenté au chapitre 2. Ce solveur offre la possibilité d'utiliser différentes stratégies de gestion des contraintes, qui sont également présentées au chapitre 2. En général, la BP donne de meilleurs résultats que la BE. Toutefois, la méthode testée est telle que les points non réalisables causent des interruptions, et les valeurs des contraintes renvoyées au solveur par la passerelle peuvent être différentes de la vérité si les interruptions surviennent à basse fidélité. La BE est une manière d'indiquer au solveur de ne pas considérer ces valeurs en ignorant les points non réalisables. Différentes impémentations mettent en pratique cette idée.

Plusieurs des implémentations sont en deux phases. Cette pratique est uniquement pertinente lorsque le point de départ n'est pas réalisable. Il y a alors une première phase de réalisabilité, qui vise à trouver un premier point réalisable. Une fois qu'un tel point est trouvé, une phase d'optimalité vise à trouver la meilleure solution possible. Cette séparation en deux phases est utile car l'algorithme 3.2 peut être nuisible durant la première phase. Lorsque le solveur calcule une valeur de $h(x)$ après une évaluation, il utilise cette valeur pour déterminer si la direction vers x à partir du meilleur point courant est une bonne direction. Or, pour tout point évalué, si les valeurs des contraintes renvoyées au solveur sont significativement différentes de la vérité car des interruptions à basse fidélité ont lieu, la valeur de $h(x)$ ne permettra peut-être pas au solveur de trouver des directions vers des points réalisables. En revanche, durant la seconde phase, plusieurs points réalisables sont généralement évalués. Lorsque le solveur utilise $f(x)$ pour déterminer si la direction vers x est une bonne direction, cette valeur de f est le résultat de la plus grande fidélité de Φ à laquelle au moins une contrainte est assignée ou d'une fidélité de 1. Les valeurs de f trouvées permettent donc de bien évaluer la qualité d'une direction. De ce fait, une implémentation en deux phases signifie qu'à la phase de réalisabilité, une seule sous-évaluation avec $\phi = 1$ est effectuée à chaque évaluation, et à la phase d'optimalité, l'algorithme 3.2 est lancé à chaque évaluation. Les implémentations testées sont les suivantes.

- Interruption BP : une seule phase avec la BP où l'algorithme 3.2 est lancé à chaque évaluation.
- Interruption BE : une seule phase avec la BE où l'algorithme 3.2 est lancé à chaque évaluation.
- Inter 2ph BP/BP : Implémentation en deux phases avec la BP.
- Inter 2ph BE/BE : Implémentation en deux phases avec la BE.
- Inter 2ph BP/BE : Implémentation en deux phases où la BP est utilisée pour la

première, et la BE est utilisée à la seconde phase. Cette méthode est différentes de la PEB présentée au chapitre 2.

5.2 Résultats numériques

La présente section est divisée en trois sous-section, une correspondant à chaque étape de l'algorithme 4.1. Pour chacune de ces étapes, les résultats pour les quatre instances SOLAR qui comportent un aspect multi-fidélité sont présentés.

5.2.1 Analyse du comportement des contraintes

L'analyse du comportement des contraintes est détaillée à la section 4.1. On y mentionne que l'utilisatuer doit choisir un ensemble Φ incluant 1, une valeur de Δ, et une taille d'hypercube latin. Lorsque les évaluations aux basses fidélités sont significativement plus rapides que les autres (ce qui est le cas pour les boîtes noires SOLAR), il est peu coûteux d'ajouter de nombreuses basses fidélités. On choisit l'ensemble

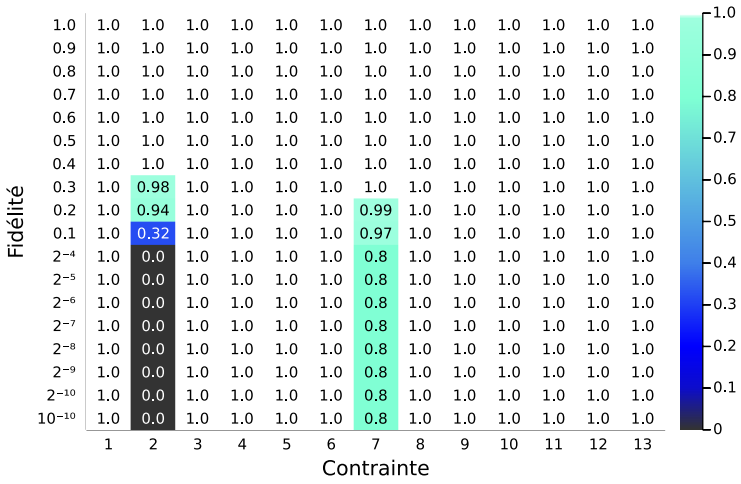
$$\Phi = \{10^{-10}, 2^{-10}, 2^{-9}, 2^{-8}, 2^{-7}, 2^{-6}, 2^{-5}, 2^{-4}, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}.$$

À des fins de comparaisons, trois différents hypercubes latins sont choisis, et ce, pour chacun des quatre problèmes SOLAR. Un total de douze instances sont donc testées, et les charactéristiques de chacune sont montrées au tableau 5.3.

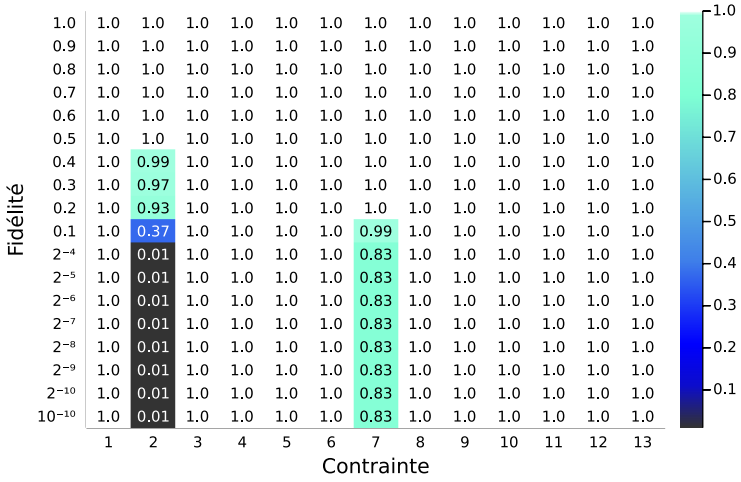
Paramètre	**Instance**											
Boîte noire	solar2			solar3			solar4			solar7		
$\lvert H\rvert$	10^4	10^3	10^3	10^4	10^3	10^3	10^4	10^3	10^3	10^4	10^3	10^3
Δ	∞	∞	$1/2$	∞	∞	$1/5$	∞	∞	$1/10$	∞	∞	$1/2$

TABLEAU 5.3 Instances testées pour l'analyse du comportement des contraintes.

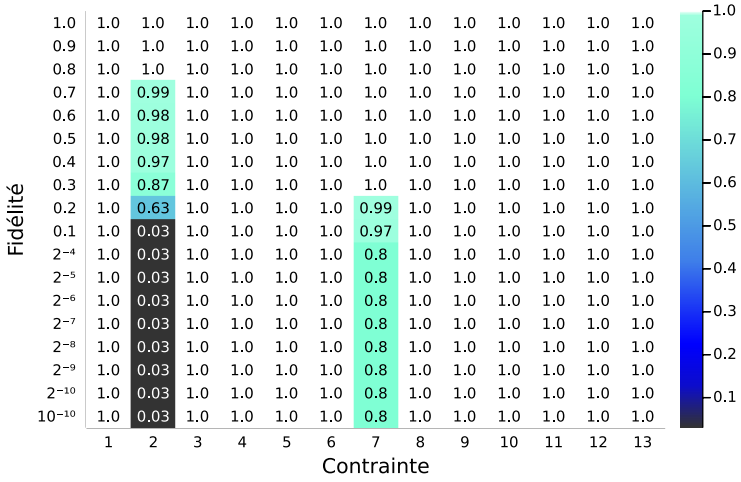
Suite aux évaluations, la valeur de p_{ij} et de r_{ij} est calculée pour chaque $(i, j) \in (I, J)$, et la valeur de t_i est calculée pour chaque fidélité dans Φ. Débutons avec les r_{ij} pour les trois instances avec solar2 montrés à la figure 5.1.



(a) Hypercube latin de 10^4 points avec $\Delta = \infty$.



(b) Hypercube latin de 10^3 points avec $\Delta = \infty$.



(c) Hypercube latin de 10^3 points avec $\Delta = 1/2$.

FIGURE 5.1 Valeurs des r_{ij} de solar2 avec différentes instances.

Les quatre contraintes affectées par la fidélité sont c_2, c_7, c_8 et c_9. Il est donc attendu que $r_{ij} = 1$ pour toute fidélité pour les autres contraintes. On observe ce comportement avec les contraintes c_8 et c_9 également. Cela signifie que pour les points échantillonnés, toutes les fidélités sont représentatives pour ces contraintes. Autrement dit, la valeur de c_8 et c_9 peut varier avec la fidélité, mais jamais de sorte à obtenir une différente valeur de β. Par exemple, le point x^0 est tel que $c_8(x^0, 10^{-10}) = -59.962561$, $c_8(x^0, 0.1) = -59.827975$ et $c_8(x^0, 1) = -59.324137$. En incluant les autres fidélités, on observe que $\beta(x^0, \phi) = 0$ pour tout $\phi \in \Phi$. Seulement les contraintes c_2 et c_7 sont telles que certaines fidélités ne sont parfois pas représentatives. Les trois instances donnent des résultats plutôt similaires, cela signifie que l'option la plus simple, un hypercube latin de 10^3 points borné par X, est suffisante.

5.2.2 Calcul de la matrice de biadjacence

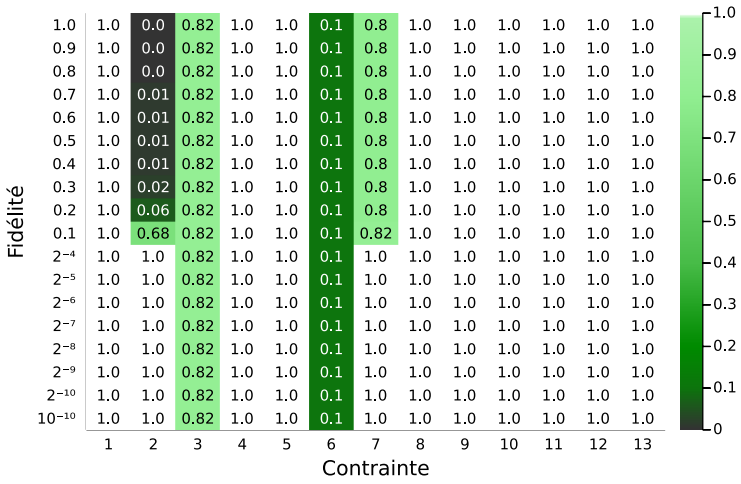
5.2.3 Optimisations avec NOMAD

Comparaison des cinq algorithmes

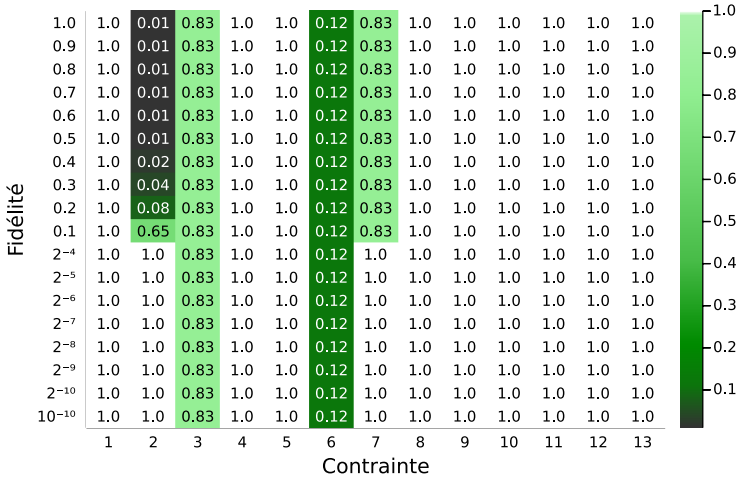
5.2.4 Comparaison des partitions

5.3 Application potentielle à PRIAD

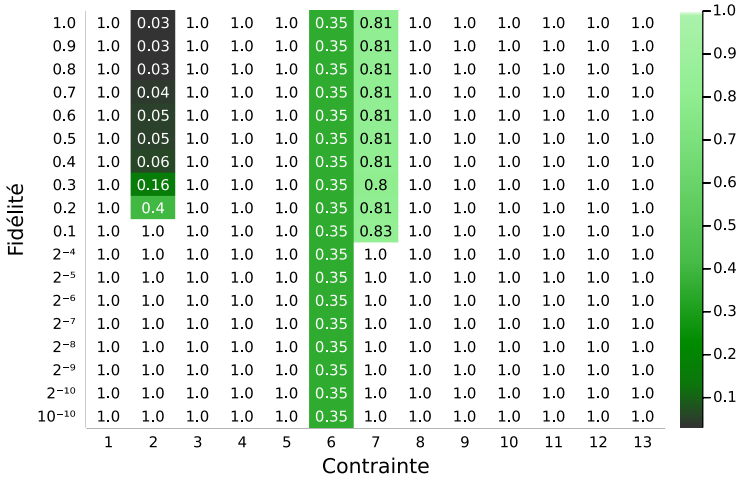
5.4 Discussion



(a) Hypercube latin de 10^4 points avec $\Delta = \infty$.



(b) Hypercube latin de 10^3 points avec $\Delta = \infty$.



(c) Hypercube latin de 10^3 points avec $\Delta = 1/2$.

FIGURE 5.2 Valeurs des r_{ij} de solar2 avec différentes instances.

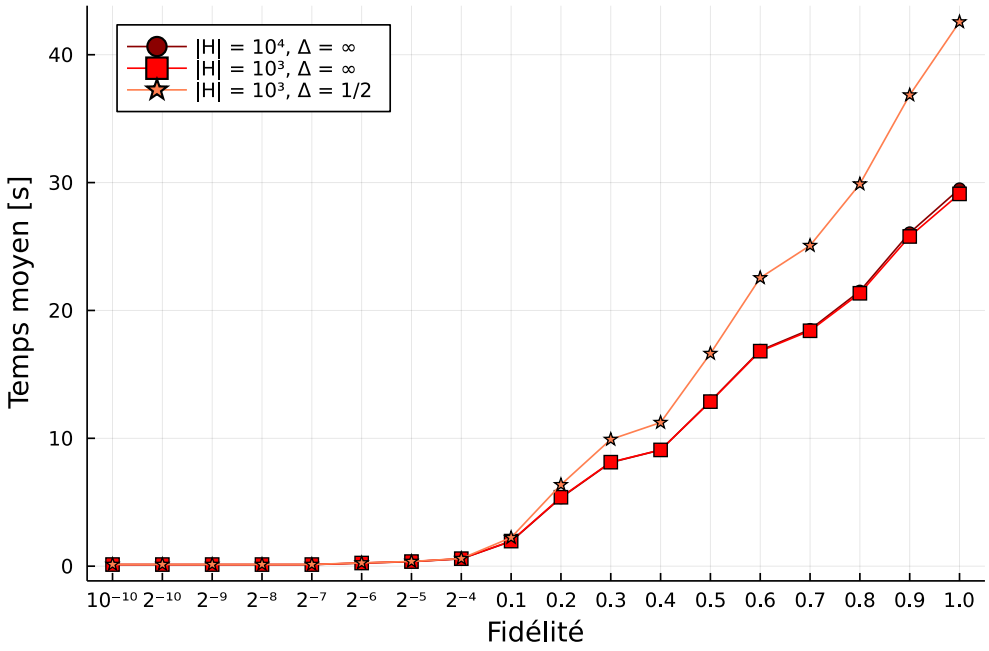


FIGURE 5.3 Valeurs des t_i de solar2 avec différentes instances.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

Rappeler conditions pour que ma méthode soit efficace : multi-fidélité, contraintes difficiles à satisfaire et représentatives à basse fidélité, et optimums proches de ces contraintes.

6.2 Limitations de la solution proposée

6.3 Améliorations futures

Pour traiter l'objectif, il serait possible d'ajouter une contrainte au niveau de la passerelle comme $f(x) \leq f^*$, et assigner cette contrainte selon une analyse du comportement de l'objectif avec la fidélité. Problème potentiel : interruptions massives, solveur n'a pas jamais de points qui indiquent une bonne direction. à tester avec autres logiciels que nomad et avec autres bb que solar.

RÉFÉRENCES

S. Alarie, N. Amaioua, C. Audet, S. Le Digabel, et L.-A. Leclaire, “Selection of variables in parallel space decomposition for the mesh adaptive direct search algorithm”, Les cahiers du GERAD, Rapp. tech. G-2018-38, 2018. En ligne : http://www.optimization-online.org/DB_HTML/2018/06/6660.html

S. Alarie, C. Audet, P.-Y. Bouchet, et S. Le Digabel, “Optimisation of stochastic blackboxes with adaptive precision”, *SIAM Journal on Optimization*, vol. 31, no. 4, pp. 3127–3156, 2021. DOI : 10.1137/20M1318894. En ligne : https://dx.doi.org/10.1137/20M1318894

S. Alarie, C. Audet, A. Gheribi, M. Kokkolaras, et S. Le Digabel, “Two decades of blackbox optimization applications”, *EURO Journal on Computational Optimization*, vol. 9, p. 100011, 2021. DOI : 10.1016/j.ejco.2021.100011. En ligne : https://doi.org/10.1016/j.ejco.2021.100011

S. Alarie, C. Audet, P. Jacquot, et S. Le Digabel, “Hierarchically constrained blackbox optimization”, *Operations Research Letters*, vol. 50, no. 5, pp. 446–451, 2022. DOI : 10.1016/j.orl.2022.06.006. En ligne : https://doi.org/10.1016/j.orl.2022.06.006

E. Anderson et M. Ferris, “A Direct Search Algorithm for Optimization with Noisy Function Evaluations”, *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 837–857, 2001. DOI : 10.1137/S1052623496312848. En ligne : https://dx.doi.org/10.1137/S1052623496312848

C. Audet et J. Dennis, Jr., “A Progressive Barrier for Derivative-Free Nonlinear Programming”, *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 445–472, 2009. DOI : 10.1137/070692662. En ligne : https://dx.doi.org/10.1137/070692662

——, “Mesh Adaptive Direct Search Algorithms for Constrained Optimization”, *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, 2006. DOI : 10.1137/040603371. En ligne : https://dx.doi.org/Doi:10.1137/040603371

C. Audet et W. Hare, *Derivative-Free and Blackbox Optimization*, série Springer Series in Operations Research and Financial Engineering. Cham, Switzerland : Springer, 2017. DOI : 10.1007/978-3-319-68913-5. En ligne : https://dx.doi.org/10.1007/978-3-319-68913-5

C. Audet, J. Dennis, Jr., et S. Le Digabel, *Parallel Space Decomposition of the Mesh Adaptive Direct Search algorithm*, série The GERAD newsletters (Eleven articles published in leading Journals), 2008, vol. 5, no. 2, p. 3. En ligne : https://www.gerad.ca/Sebastien.Le.Digabel/Publications/bulletin_gerad_2009-01-22.pdf

——, "Globalization strategies for Mesh Adaptive Direct Search", *Computational Optimization and Applications*, vol. 46, no. 2, pp. 193–215, 2010. DOI : 10.1007/s10589-009-9266-1. En ligne : https://dx.doi.org/10.1007/s10589-009-9266-1

C. Audet, A. Conn, S. Le Digabel, et M. Peyrega, "A progressive barrier derivative-free trust-region algorithm for constrained optimization", *Computational Optimization and Applications*, vol. 71, no. 2, pp. 307–329, 2018. DOI : 10.1007/s10589-018-0020-4. En ligne : https://dx.doi.org/10.1007/s10589-018-0020-4

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*, vol. 12, no. 4, pp. 675–689, 2018. DOI : 10.1007/s11590-017-1226-6. En ligne : https://dx.doi.org/10.1007/s11590-017-1226-6

C. Audet, K. Dzahini, M. Kokkolaras, et S. Le Digabel, "Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates", *Computational Optimization and Applications*, vol. 79, no. 1, pp. 1–34, 2021. DOI : 10.1007/s10589-020-00249-0. En ligne : https://doi.org/10.1007/s10589-020-00249-0

C. Audet, A. Batailly, et S. Kojtych, "Escaping Unknown Discontinuous Regions in Blackbox Optimization", *SIAM Journal on Optimization*, vol. 32, no. 3, pp. 1843–1870, 2022. DOI : 10.1137/21M1420915. En ligne : https://doi.org/10.1137/21M1420915

C. Audet, S. Le Digabel, V. Rochon Montplaisir, et C. Tribes, "Algorithm 1027 : NOMAD version 4 : Nonlinear optimization with the MADS algorithm", *ACM Transactions on Mathematical Software*, vol. 48, no. 3, pp. 35 :1–35 :22, 2022. DOI : 10.1145/3544489. En ligne : https://dx.doi.org/10.1145/3544489

C. Audet, S. Le Digabel, L. Salomon, et C.Tribes, "Constrained Blackbox Optimization with the NOMAD Solver on the COCO Constrained Test Suite", dans *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, série GECCO '22. New York, NY, USA : Association for Computing Machinery, 2022, pp. 1683–1690. DOI : 10.1145/3520304.3534019. En ligne : https://doi.org/10.1145/3520304.3534019

I. Bajaj, S. Iyer, et M. Faruque Hasan, “A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point”, *Computers & Chemical Engineering*, vol. 116, pp. 306–321, 2018, multi-scale Systems Engineering – in memory & honor of Professor C.A. Floudas. DOI : https://doi.org/10.1016/j.compchemeng.2017.12.011. En ligne : https://www.sciencedirect.com/science/article/pii/S0098135417304404

V. Balabanov et G. Venter, “Multi-fidelity optimization with high-fidelity analysis and low-fidelity gradients”, dans *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2004, p. 4459.

R. Barton et J. Ivey, Jr., “Nelder-Mead simplex modifications for simulation optimization”, *Management Science*, vol. 42, no. 7, pp. 954–973, 1996. DOI : 10.1287/mnsc.42.7.954. En ligne : https://dx.doi.org/10.1287/mnsc.42.7.954

S. Belakaria, A. Deshwal, et J. Doppa, “Multi-Fidelity Multi-Objective Bayesian Optimization : An Output Space Entropy Search Approach”, *CoRR*, vol. abs/2011.01542, 2020. En ligne : https://arxiv.org/abs/2011.01542

K. Chang, “Stochastic nelder-mead simplex method - a new globally convergent direct search method for simulation optimization”, *European Journal of Operational Research*, vol. 220, no. 3, pp. 684–694, 2012. DOI : 10.1016/j.ejor.2012.02.028. En ligne : https://dx.doi.org/10.1016/j.ejor.2012.02.028

X. Chen et C. Kelley, “Optimization with hidden constraints and embedded Monte Carlo computations”, *Optimization and Engineering*, vol. 17, no. 1, pp. 157–175, 2016. DOI : 10.1007/s11081-015-9302-1. En ligne : https://dx.doi.org/10.1007/s11081-015-9302-1

A. Conn, K. Scheinberg, et P. Toint, “On the convergence of derivative-free methods for unconstrained optimization”, dans *Approximation Theory and Optimization : Tributes to M.J.D. Powell*, M. Buhmann et A. Iserles, éds. Cambridge, United Kingdom : Cambridge University Press, 1997, pp. 83–108. En ligne : http://us.cambridge.org/Titles/catalogue.asp?isbn=0521581907

A. Côté, O. Blancke, S. Alarie, A. Dems, D. Komljenovic, et D. Messaoudi, “Combining Historical Data and Domain Expert Knowledge Using Optimization to Model Electrical Equipment Reliability”, dans *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, Liege, Belgium,

2020, pp. 1–6. DOI : 10.1109/PMAPS47429.2020.9183620. En ligne : https://dx.doi.org/10.1109/PMAPS47429.2020.9183620

K. Dzahini, M. Kokkolaras, et S. Le Digabel, “Constrained stochastic blackbox optimization using a progressive barrier and probabilistic estimates”, Les cahiers du GERAD, Rapp. tech. G-2020-60, 2022, to appear in *Mathematical Programming*. DOI : 10.1007/s10107-022-01787-7. En ligne : http://www.optimization-online.org/DB_HTML/2020/11/8101.html

M. Gaha, B.Chabane, D.Komljenovic, A. Côté, C. Hébert, O. Blancke, A. Delavari, et G. Abdul-Nour, “Global Methodology for Electrical Utilities Maintenance Assessment Based on Risk-Informed Decision Making”, *Sustainability*, vol. 13, no. 16, p. 9091, 2021.

A. Galántai, “Convergence of the Nelder-Mead method”, *Numerical Algorithms*, vol. 90, no. 3, pp. 1043–1072, 2022.

N. Gould et P. Toint, “Nonlinear programming without a penalty function or a filter”, *Mathematical Programming*, vol. 122, no. 1, pp. 155–196, 2010. DOI : 10.1007/s10107-008-0244-7. En ligne : https://dx.doi.org/10.1007/s10107-008-0244-7

N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, et D. Brockhoff, “COCO : a platform for comparing continuous optimizers in a black-box setting”, *Optimization Methods and Software*, vol. 36, no. 1, pp. 114–144, 2021. DOI : 10.1080/10556788.2020.1808977. En ligne : https://doi.org/10.1080/10556788.2020.1808977

P.-L. Huot, A. Poulin, C. Audet, et S. Alarie, “A hybrid optimization approach for efficient calibration of computationally intensive hydrological models”, *Hydrological Sciences Journal*, vol. 64, no. 10, pp. 1204–1222, 2019. DOI : 10.1080/02626667.2019.1624922. En ligne : https://dx.doi.org/10.1080/02626667.2019.1624922

W. Huyer et A. Neumaier, “Global optimization by multilevel coordinate search”, *Journal of Global Optimization*, vol. 14, pp. 331–355, 1999.

D. Komljenovic, D. Messaoudi, A. Côté, M. Gaha, L. Vouligny, S. Alarie, et O. Blancke, “Asset Management in Electrical Utilities in the Context of Business and Operational Complexity”, dans *World Congress on Resilience, Reliability and Asset Management*, Singapore, 2019, pp. 148–151.

S. Le Digabel, “Algorithm 909 : NOMAD : Nonlinear Optimization with the

MADS algorithm", *ACM Transactions on Mathematical Software*, vol. 37, no. 4, pp. 44 :1–44 :15, 2011. DOI : 10.1145/1916461.1916468. En ligne : https://dx.doi.org/10.1145/1916461.1916468

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

M. Lemyre Garneau, "Modelling of a solar thermal power plant for benchmarking blackbox optimization solvers", Mémoire de maîtrise, Polytechnique Montréal, 2015. En ligne : https://publications.polymtl.ca/1996/

J. Müller, "MISO : mixed-integer surrogate optimization framework", *Optimization and Engineering*, vol. 17, no. 1, pp. 177–203, 2016. DOI : 10.1007/s11081-015-9281-2. En ligne : https://dx.doi.org/10.1007/s11081-015-9281-2

K. Murphy, C. Carter, et L. Wolfe, "How long should I simulate, and for how many trials ? A practical guide to reliability simulations", dans *Annual Reliability and Maintainability Symposium. 2001 Proceedings. International Symposium on Product Quality and Integrity (Cat. No. 01CH37179)*. IEEE, 2001, pp. 207–212.

K. Murphy, A. Malerich, et C. Carter, "What is truth ? A practical guide to comparing reliability equation answers to simulation results", dans *Annual Reliability and Maintainability Symposium, 2005. Proceedings.* IEEE, 2005, pp. 439–444.

J. Nelder et R. Mead, "A simplex method for function minimization", *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965. DOI : 10.1093/comjnl/7.4.308. En ligne : http://comjnl.oxfordJournals.org/content/7/4/308.abstract

Y. Ozaki, S. Watanabe, et M. Onishi, "Accelerating the Nelder-Mead method with predictive parallel evaluation", dans *6th ICML Workshop on Automated Machine Learning*, vol. 185, 2019, p. 186.

T. Papalexopoulos, C. Tjandraatmadja, R. Anderson, J. Vielma, et D. Belanger, "Constrained Discrete Black-Box Optimization using Mixed-Integer Programming", dans *Proceedings of the 39th International Conference on Machine Learning*, série Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, et S. Sabato, éds., vol. 162. PMLR, 17–23 Jul 2022, pp. 17 295–17 322. En ligne : https://proceedings.mlr.press/v162/papalexopoulos22a.html

N. Ploskas et N. Sahinidis, “Review and comparison of algorithms and software for mixed-integer derivative-free optimization”, *Journal of Global Optimization*, vol. 82, no. 3, pp. 433–462, 2022. DOI : 10.1007/s10898-021-01085-0. En ligne : https://dx.doi.org/10.1007/s10898-021-01085-0

S. Razavi, B. Tolson, L. Matott, N. Thomson, A. MacLean, et F. Seglenieks, “Reducing the computational cost of automatic calibration through model preemption”, *Water Resources Research*, vol. 46, no. 11, 2010.

L. Rios et N. Sahinidis, “Derivative-free optimization : a review of algorithms and comparison of software implementations”, *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013. DOI : 10.1007/s10898-012-9951-y. En ligne : https://dx.doi.org/10.1007/s10898-012-9951-y

R. Sen, K. Kandasamy, et S. Shakkottai, “Multi-fidelity black-box optimization with hierarchical partitions”, dans *International conference on machine learning*. PMLR, 2018, pp. 4538–4547.

V. Torczon, “On the convergence of pattern search algorithms”, *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997. DOI : 10.1137/S1052623493250780. En ligne : https://dx.doi.org/10.1137/S1052623493250780

J. Wang, D. Basu, et I. Trummer, “Procrastinated Tree Search : Black-Box Optimization with Delayed, Noisy, and Multi-Fidelity Feedback”, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, pp. 10 381–10 390, Jun. 2022. DOI : 10.1609/aaai.v36i9.21280. En ligne : https://ojs.aaai.org/index.php/AAAI/article/view/21280

M. Wetter et E. Polak, “Building design optimization using a convergent pattern search algorithm with adaptive precision simulations”, *Energy and Buildings*, vol. 37, no. 6, pp. 603–612, 2005.

J. Wu, S. Toscano-Palmerin, P. Frazier, et A. Wilson, “Practical Multi-fidelity Bayesian Optimization for Hyperparameter Tuning”, dans *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, série Proceedings of Machine Learning Research, R. P. Adams et V. Gogate, éds., vol. 115. PMLR, 22–25 Jul 2020, pp. 788–798. En ligne : https://proceedings.mlr.press/v115/wu20a.html

ANNEXE A DÉMO

ANNEXE B ENCORE UNE ANNEXE

Texte de l'annexe B en mode «landscape».

ANNEXE C UNE DERNIÈRE ANNEXE

Texte de l'annexe C.