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 pour partitionner les contraintes, où chaque partie est couplée à une fidélité. Les parties sont hiérarchisées selon les valeurs des fidélités couplées. Des travaux futurs pourront étudier la possibilité d'agencer

la méthode de hiérarchie 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 partition des 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 ainsi que les méthodes de réduction du temps d'optimisation 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. Les hypothèses les plus faibles possibles sont alors posées sur la fonction objectif et sur les 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 et supérieure 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.

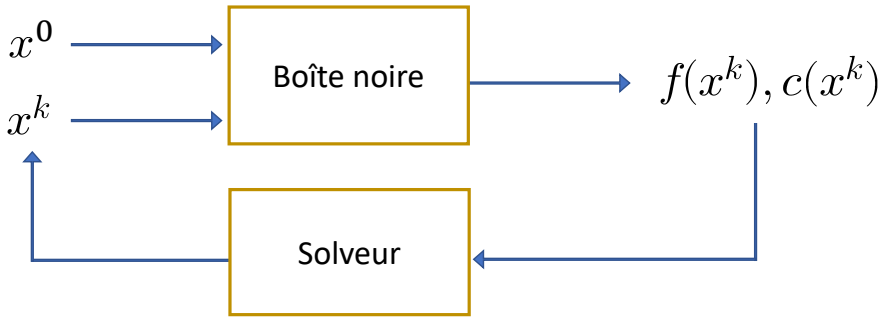

FIGURE 2.1 Généralisation des méthodes d'optimisation de boîte noire

Les algorithmes de BBO sont divisés en trois grandes catégories. La première consiste en les méthodes qui génèrent des substitus de la boîte noire. Un substitut est un modèle qui tente de répliquer le comportement de la boîte noire, en ayant un coût d'évaluation faible. Le cadre d'optimisation de modèles 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. 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).

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

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 d'utiliser cette information pour potentiellement trouver de meilleurs points réalisables tout

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 $\min_{x \in X} f_\Omega$

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.

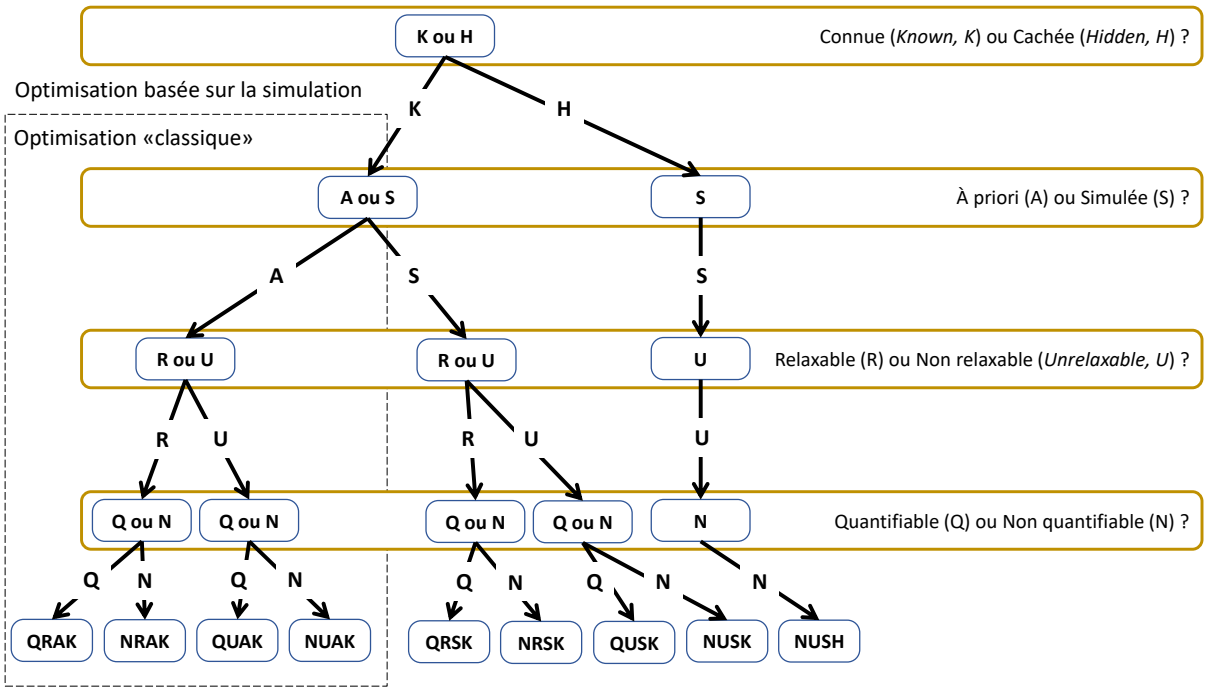

FIGURE 2.2 Représentation en arbre de la taxonomie des contraintes QRAK traduite de Le Digabel et Wild (2015)

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 modèle 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).

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
$\lambda : [0, 1] \to \mathbb{R}^+$	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 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 est définie comme le coût à des fins de généralité, or, pour ce projet de maîtrise, cette variable correspond toujours 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, 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 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.

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 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

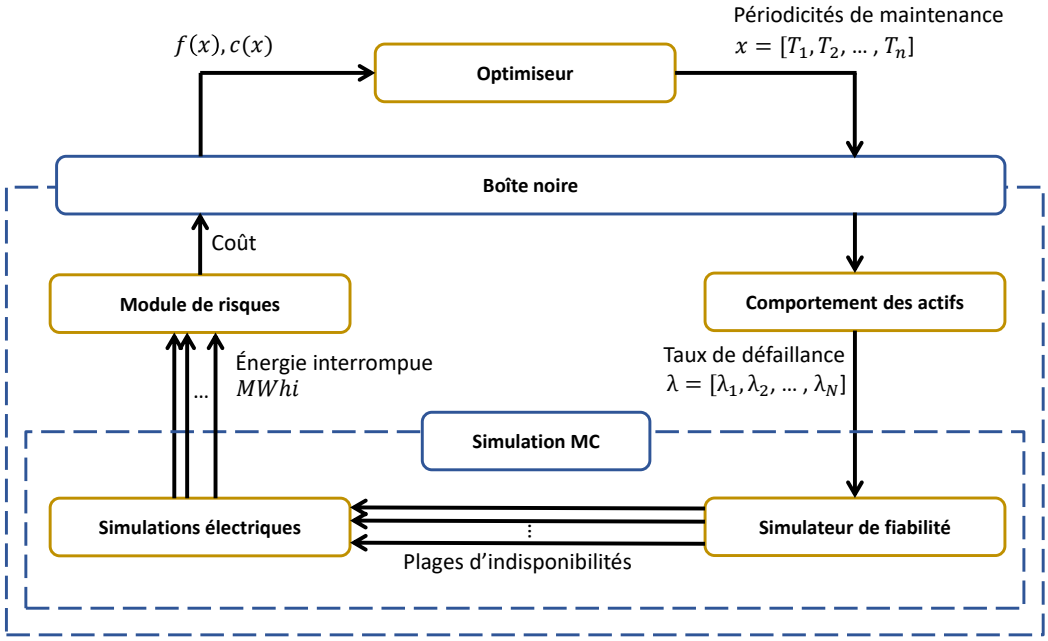

FIGURE 2.3 Représentation graphique du simulateur de PRIAD

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 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.

CHAPITRE 3 SOUS-ÉVALUATIONS INTERROMPUES

Ce chapitre décrit dans une première sous-section la méthode proposée d'un point de vue haut niveau. L'algorithme 3.1 décrit cette méthode en trois étapes. Les deux premières seront détaillées au chapitre 4, alors que le présent chapitre met l'accent sur la dernière étape : celle de l'optimisation de la boîte noire. La première sous-section établit également certaines définitions. En effet, comme cette méthode est originale, quelques nouvelles définitions qui sont utilisées dans le reste de ce mémoire sont données. Dans la seconde sous-section, l'étape trois est détaillée rigoureusement.

3.1 Description de la méthode et définitions

Tel que mentionné à l'introduction, la méthode proposée consiste à partitionner les contraintes d'un problème d'optimisation de boîte noire en parties auxquelles sont couplées des fidélités. Les parties sont hiérarchisées selon les valeurs des fidélités couplées. Une partition hiérarchisée doit 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. Cette partition hiérarchisée est ensuite utilisée par l'algorithme des sous-évaluations interrompues décrit par l'algorithme 3.2 lors d'une optimisation. Cet algorithme vise à lancer la boîte noire séquentiellement aux différentes fidélités en suivant la hiérarchie, en ne considérant que les contraintes dont la valeur est pertinente à la fidélité évaluée. Après chaque appel, si une contrainte n'est pas satisfaite, l'algorithme se termine. 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.

Algorithme 3.1 : Optimisation avec partition des contraintes hiérarchisée

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. Calculer une partition optimale
3. Lancer l'optimisation avec un solveur et l'algorithme 3.2

Il est nécessaire que la partition soit déterminée en fonction du comportement des contraintes à différentes fidélités pour qu'elle permette à l'algorithme de sous-évaluations interrompues

d'effectuer des interruptions justes. La première étape consiste alors à effectuer une analyse de ces comportements. La seconde étape se base sur cette analyse pour calculer une partition optimale des contraintes, et finalement l'optimisation est lancée en suivant l'algorithme de sous-évaluations interrompues. L'algorithme 3.1 illustre ces trois étapes d'un point de vue haut niveau.

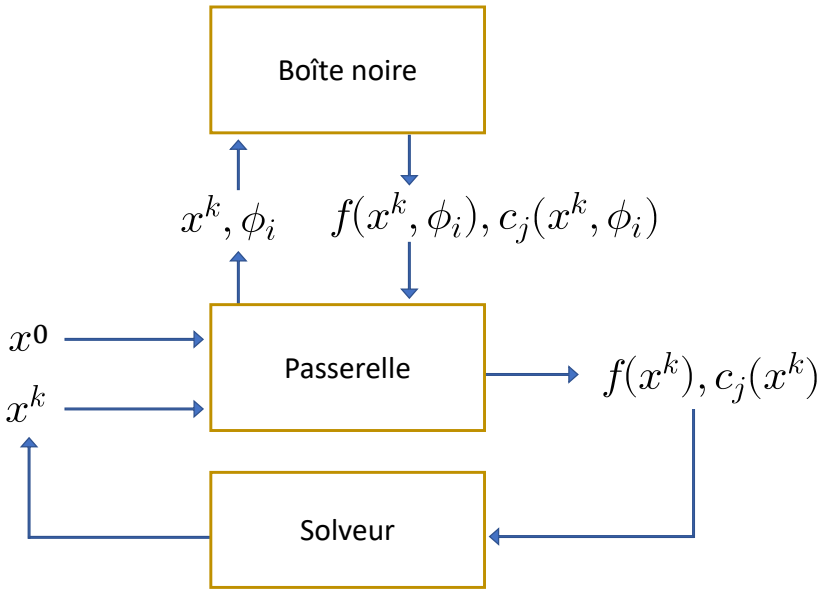

FIGURE 3.1 Boucle d'optimisation avec un solveur et la passerelle

L'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 de l'algorithme de sous-évaluations interrompues. 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. Avant de détailler l'algorithme, quelques nouveaux termes sont définis au tableau 3.1.

La première définition du tableau 3.1 est une adaptation de la première définition du tableau 2.2 aux contraintes. En effet, les méthodes d'optimisation de boîte noire multi-fidélité ou stochastique présentées à la section 2 considèrent uniquement le cas non contraint. Seulement la valeur de l'objectif en fonction de x et de la fidélité a donc été définie. Si la multi-fidélité est le résultat d'un procéssus stochastique, il est possible que $c(x, \phi)$ varie d'une évaluation

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
$\ell \in \mathbb{R}$	Nombre fini de fidélités choisies par l'utilisateur.
$\Phi \in [0, 1]^\ell$	Ensemble des fidélités choisies par l'utilisateur.
ϕ_i	i-ième fidélité de Φ, où $i \in I := \{1, 2, \ldots, \ell\}$.
$\chi \in \mathbb{B}^{\ell x m}$	Matrice exprimant une partition des contraintes.

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

à l'autre pour le même x, et c ne serait donc pas une fonction. Ce constat est également valide pour $f(x, \phi)$. Concernant ℓ, 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 ($\ell \neq \infty$), l'utilisateur doit alors choisir un ensemble de taille fini Φ. Le choix de la taille de Φ dépend de la quantité de parallélisme disponible à l'utilisateur. Le chapitre 4 propose d'inclure un filtrage des fidélités non nécessaires à l'étape deux de l'algorithme 3.1. Il peut donc sembler avantageux d'inclure un grand nombre de fidélités, mais plus ce nombre est grand, plus l'étape un de l'algorithme prendra du temps. Le chapitre 4 propose également une méthode pour l'étape 1, où plusieurs tâches indépendantes doivent êtres effectuées. Celles-ci peuvent être lancées en parallèle, ce qui réduit ce temps. Somme toute, plus la capacité de calcul parallèle de l'utilisateur est grande, plus il peut se permettre un grand ℓ. La dernière définition du tableau attribue une variable à la partition des contraintes. La i-ème rangée de la matrice χ représente la partie couplée à ϕ_i, et chaque colonne de la matrice représente une contrainte. Dans le but d'alléger le texte, l'abus de langage « la contrainte c_j est partitionnée à ϕ_i » est utilisé dans ce mémoire pour signifier « selon la partition, la contrainte c_j appartient à la partie à laquelle est couplée la fidélité ϕ_i ». On a donc la définition donnée par (3.1).

$$\chi_{ij} := \begin{cases} 1 \text{ si } c_j \text{ est partitionnée à } \phi_i \\ 0 \text{ sinon} \end{cases} \tag{3.1}$$

Pour illustrer ces propos, la figure 3.2 montre un exemple de problème où $m = 5$ et $\ell = 4$. À gauche est montrée une partition hiérarchisée sous forme matricielle et à droite est montrée une représentation graphique de la partition et du couplage des fidélités aux parties.

$$\chi = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \equiv$$

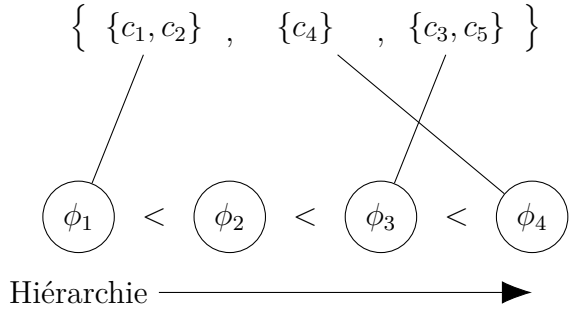

FIGURE 3.2 Exemple de partition des contraintes avec un couplage aux fidélités.

3.2 Algorithme de sous-évaluations interrompues

L'algorithme de sous-évaluations interrompues est décrit rigoureusement à l'algorithme 3.2. Ce dernier est lancé par la passerelle à chaque fois qu'elle reçoit un point à évaluer du solveur (ou le point de départ). L'algorithme lance plusieurs fois la boîte noire à différentes fidélités. Ces lancements sont appelés des sous-évaluations, de manière à ce que le terme évaluation soit relatif au point de vue du solveur. Chaque évaluation constitue un achèvement de l'algorithme 3.2, qui lui effectue plusieurs sous-évaluations.

Chaque fois que la passerelle reçoit un point x^k du solveur, l'algorithme effectue une première sous-évaluation à la plus petite fidélité à laquelle au moins une contrainte est partitionnée. Après la sous-évaluation, les valeurs des contraintes partitionnées à cette fidélité sont observées. Si elles sont satisfaites, une sous-évaluation à la prochaine plus petite fidélité à laquelle au moins une contrainte est partitionnée est effectuée. Cette fois, les valeurs des contraintes partitionnées à cette fidélité et aux fidélités plus basses sont observées. Si elles sont toutes satisfaites, l'algorithme poursuit. Les parties sont considérées en ordre croissant de fidélité, d'où le concept de parties hiérarchisées. Dès qu'une sous-évaluation résulte en au moins une contrainte observée non satisfaite, une interruption est effectuée. L'algorithme se termine et les sorties les plus récentes sont envoyées au solveur. Si x^k est réalisable, aucune interruption ne sera effectuée. Le cas échéant, une dernière étape est suivie. Soit x^* le point réalisable tel que $f(x^*)$ est la plus petite parmi les évaluations effectuées durant l'optimisation. Si x^k est réalisable à la dernière sous-évaluation, résulte en un meilleur objectif que x^* et que la dernière fidélité n'est pas 1 (la vérité), une dernière sous-évaluation est effectuée à $\phi = 1$. L'implémentation de l'algorithme doit donc inclure une méthode pour garder en mémoire $f(x^*)$.

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 $\mathit{Exécution}(x, \phi)$ dénote un lan-

cement de la boîte noire, et elle renvoie $f(x, \phi)$ et $c(x, \phi)$. L'expression e_i dénote le vecteur de longeur m qui ne contient que des 0 à l'exception du i-ième élément qui vaut 1. Une inégalité posée sur un vecteur est vraie si chaque élément du vecteur la satisfait individuellement.

Algorithme 3.2 : Sous-évaluations interrompues

Étant donné une fonction boîte noire $Exécution : (\mathbb{R}^n, [0, 1]) \to (\mathbb{R} \cup \{\infty\}, (\mathbb{R} \cup \{\infty\})^m)$, un ensemble Φ, une partition χ, un point x et $f(x^*)$.

$i = 1$

$f, c = Exécution(x, \phi_i)$

Tant que $\{c_j \in c : \chi_{aj} = 1\} \leq 0 \, \forall a \in I \leq i$ **et** $i < \ell$ **faire**

  $i \leftarrow i + 1$

  Si non$(e_i^\top \chi = 0)$

    $f, c \leftarrow Exécution(x, \phi_i)$

Si $i = \ell$, $c \leq 0$, $f < f(x^*)$ et $1 \notin \Phi$

  $f, c \leftarrow Exécution(x, 1)$

Renvoyer f,c

Dans la partition hiérarchisée de la figure 3.2, c_1 et c_2 sont partitionnées à ϕ_1. 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 seraient 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 $e_2^\top \chi = [0, 0, 0, 0, 0]$. L'algorithme ne lance donc pas de sous-évaluation avec ϕ_2. Les contraintes c_3, c_4 et c_5 sont partitionnées à des fidélités plus coûteuses. Elles ne permetteront pas de sauver autant de temps. De plus, si c_1, c_2, c_3 et c_5 sont toutes satisfaites à toutes les sous-évaluations, l'achèvement de l'algorithme peut prendre plus de temps que si une seule sous-évaluation à la plus grande fidélité était directement effectuée. En revanche, si c_4 ne donne de l'information pertinente qu'à ϕ_4, partitionner 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 partition de qualité est une partition qui permet d'identifier avec justesse si certaines contraintes sont satisfaites ou non en vérité à

partir d'informations à différentes fidélités. Aussi, la quantitié 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.

Pour terminer avec l'exemple de la figure 3.2, si un x est tel que toutes les contraintes sont satisfaites à toutes les sous-évaluations, i est incrémenté jusqu'à ℓ et $c \leq 0$. Certaines autres conditions sont alors vérifiées avant de potentiellement lancer une dernière sous-évaluation à $\phi = 1$. L'intérêt de cette sous-évaluation est que si $\phi_4 \neq 1$, il est possible que la valeur de l'objectif ne réflète pas la vérité. Dans le cas où x deviendra le nouveau x^* car $f < f(x^*)$, il est possible que le solveur atteigne un critère d'arrêt et que la solution renvoyée à l'utilisateur soit x. L'ajout de l'évaluation à $\phi = 1$ garanti que lorsque l'optimisation sera terminée, la solution est véritablement la meilleure parmi les points évalués. De plus, il est possible que x soit réalisable à ϕ_4, mais ne le soit pas à $\phi = 1$. Cela est rare si la partition est de grande qualité, mais il est prioritaire de garantir qu'un point qui pourrait être renvoyé à l'utilisateur en tant que meilleure solution réalisable soit véritablement réalisable.

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 recevera 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 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 porpriétés 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 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 parmis les points de Ω, plus il y a de temps à réduire.

CHAPITRE 4 CALCUL DE LA PARTITION DES CONTRAINTES

4.1 Représentativité des contraintes

define ε

4.2 Satisfaction des contraintes

4.3 Temps d'évaluation

4.4 Modèle d'optimisation de la partition

Montrer modèle en remplacants les y_i par $min\{1, \sum_{j=1}^{m} \chi_{ij}\}$, puis mentionner que min remplacable par la contrainte ....

$$\min \quad \mathbb{E}\left[\text{ temps d'une évaluation }\right] \tag{4.1}$$

$$= \min_{\chi \in \mathbb{B}^{\ell x m}} \quad t_1 \min\left\{1, \sum_{j=1}^{m} \chi_{1j}\right\} + \sum_{i=2}^{\ell}\left(t_i \min\left\{1, \sum_{j=1}^{m} \chi_{ij}\right\} \prod_{k=1}^{i-1} \prod_{j \in J} p_{kj}\chi_{kj} + 1 - \chi_{kj}\right) \tag{4.2}$$

s.c.

$$\sum_{i \in I} \chi_{ij} = 1 \quad \forall\, j \in J \tag{4.3}$$

$$r_{ij} \geq (1-\varepsilon)\chi_{ij} \quad \forall\, i \in I, \forall\, j \in J \tag{4.4}$$

Définition A.1

$$\hat{I} := \bigcup_{j \in J} \min\{i \in I : r_{ij} \geq 1 - \varepsilon\} \tag{4.5}$$

Propriété A.1

$$r_{aj} \geq 1 - \varepsilon \implies r_{bj} \geq 1 - \varepsilon \quad \forall j \in J, \forall \text{ paire } a, b \in I : b \geq a \tag{4.6}$$

Lemme A.1

Proposition

Soit χ^b une solution réalisable telle que au moins une c_j est partitionnée à $\phi_{i'}$ où $i' \in I$ et $i' \notin \hat{I}$, et χ^a une solution identique à l'exception que $\chi^a_{i'j} = 0$ et $\chi^a_{i'-1\,j} = 1 \forall j \in J : \chi^b_{i'j} = 1$. χ^a est également réalisable.

Démonstration

χ^a respecte (4.4) :

$$\begin{aligned}
\chi^b \text{ est réalisable} &\implies r_{i'j} \geq 1-\varepsilon \quad \forall j \in J : \chi^b_{i'j} = 1 \\
(4.6) &\implies i' \geq \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J : \chi^b_{i'j} = 1 \\
i' \notin \hat{I} &\implies i' > \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J : \chi^b_{i'j} = 1 \\
&\implies i'-1 \geq \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J : \chi^b_{i'j} = 1 \\
(4.6) &\implies r_{i'-1\,j} \geq 1-\varepsilon \quad \forall j \in J : \chi^b_{i'j} = 1 \\
&\implies r_{i'-1\,j} \geq 1-\varepsilon \quad \forall j \in J : \chi^a_{i'-1\,j} = 1 \\
\chi^a_{ij} = \chi^b_{ij}\, \forall i \in I\,, \forall j \in J : \chi^b_{i'j} = 0 &\implies (4.4) \text{ est respectée par } \chi^a \qquad (4.7)
\end{aligned}$$

χ^a respecte (4.3) :

$$\begin{aligned}
\chi^b \text{ est réalisable} &\implies \sum_{i \in I} \chi^b_{ij} = 1 \quad \forall\, j \in J \\
&\implies \sum_{i=1}^{i'-2} \chi^b_{ij} + 0 + 1 + \sum_{i=i'+1}^{l} \chi^b_{ij} = 1 \quad \forall j \in J : \chi^b_{i'j} = 1 \\
&\implies \sum_{i=1}^{i'-2} \chi^a_{ij} + 1 + 0 + \sum_{i=i'+1}^{l} \chi^a_{ij} = 1 \quad \forall j \in J : \chi^a_{i'-1\,j} = 1 \\
&\implies \sum_{i \in I} \chi^a_{ij} = 1 \quad \forall\, j \in J \\
&\implies (4.3) \text{ est respectée par } \chi^a \qquad (4.8)
\end{aligned}$$

$$(4.7) \text{ et } (4.8) \implies \chi^a \text{ est réalisable} \qquad (4.9)$$

□

Lemme A.2

Proposition

Soit χ^b une solution réalisable telle que au moins une c_j est partitionnée à $\phi_{i'}$ où $i' \in I$ et $i' \notin \hat{I}$, et χ^a une solution identique à l'exception que $\chi^a_{i'j} = 0$ et $\chi^a_{i'-1\,j} = 1\,\forall j \in J : \chi^b_{i'j} = 1$. $f(\chi^b) \geq f(\chi^a)$.

Démonstration

Adoptons d'abord les trois définitions suivantes qui simplifieront la notation. Étant donné une solution χ^s,

$$y^s_i = \min\left\{1, \sum_{j=1}^{m} \chi^s_{ij}\right\},$$

$$\overset{i}{\underset{k=1}{M}}{}^{s} := y^s_i \prod_{k=1}^{i-1} \prod_{j\in J} p_j \cdot \chi^s_{kj} + 1 - \chi^s_{kj}.$$

Deux scénarios peuvent survenir, dépendemment de la valeur de $y^b_{i'-1}$.

Si $y^b_{i'-1} = 0$:

$$\begin{aligned}
f(\chi^b) &= t_1 y^b_1 + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{b} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{b} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{b} \\
&= t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{b} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{a} \\
&\qquad\qquad \text{car } \chi^b_{ij} = \chi^a_{ij},\ \forall i \notin \{i'-1, i'\},\ \forall j \in J \\
&= t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} + t_{i'} \overset{i'-1}{\underset{k=1}{M}}{}^{a} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{a} \\
&\qquad\qquad \text{car } \overset{i'}{\underset{k=1}{M}}{}^{b} = \overset{i'-1}{\underset{k=1}{M}}{}^{a} \\
&\geq t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{a} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{a} \\
&\qquad\qquad \text{car } t_{i'-1} \leq t_{i'} \text{ car 2.2} \\
&= t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{a} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{a} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{a} \\
&\qquad\qquad \text{car } t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{a} = t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} = 0 \\
&= f(\chi^a) \qquad (4.10)
\end{aligned}$$

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

$$
\begin{aligned}
f(\chi^b) =& t_1 y_1^b + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{b} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{b} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{b} \\
=& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{b} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{a} \\
& \qquad \text{car } \chi^b_{ij} = \chi^a_{ij},\ \forall i \notin \{i'-1, i'\},\ \forall j \in J \\
=& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i y_i^a \overset{i-1}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{a} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{b} + \sum_{i=i'+1}^{l} t_i y_i^a \overset{l-1}{\underset{k=1}{M}}{}^{a} \\
& \qquad \text{car } t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{b} = t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{a} \\
>& t_1 y_1^a + \sum_{i=2}^{i'-2} t_i \overset{i}{\underset{k=1}{M}}{}^{a} + t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{a} + t_{i'-1} \overset{i'-1}{\underset{k=1}{M}}{}^{a} + \sum_{i=i'+1}^{l} t_i \overset{l}{\underset{k=1}{M}}{}^{a} \\
& \qquad \text{car } t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{b} > 0 = t_{i'} \overset{i'}{\underset{k=1}{M}}{}^{a} \\
=& f(\chi^a)
\end{aligned} \tag{4.11}
$$

$$
\text{(4.10) et (4.11)} \implies f(\chi^a) \leq f(\chi^b) \tag{4.12}
$$

□

Théorème 1

Proposition

À optimalité, les ϕ_i auxquelles des contraintes sont partitionnées sont telles que $i \in \hat{I}$.

Démonstration

Les Lemmes A.1 (4.9) et A.2 (4.12) impliquent que pour une solution réalisable donnée où $\exists i' \notin \hat{I} : \sum_{j \in J} \chi_{i'j} > 0$, il existe une autre solution réalisable égale ou meilleure. Une solution est optimale s'il n'existe pas de meilleure solution réalisable.

$$
\begin{aligned}
&\implies \forall\, \chi^* \in \underset{\chi}{\operatorname{argmin}} \{f(x) : x \in \Omega\},\ \nexists i' \notin \hat{I} : \sum_{j \in J} \chi_{i'j} > 0 \\
&\implies \forall\, \chi^* \in \underset{\chi}{\operatorname{argmin}} \{f(x) : x \in \Omega\},\ \forall i : \sum_{j \in J} \chi_{ij} \geq 1,\ i \in \hat{I}
\end{aligned} \tag{4.13}
$$

□

4.5 Dépendance entre la satisfaction des contraintes et la fidélité

4.6 Réductions de la taille du modèle

in preuve : avec λ du tableau 2.2, optimalité garantie du modèle d'opt analytique. Sans, il peut exister une haute fidélité pas rapport qui a un temps super bas où il faut partitionner ds contraintes pour partition optimale, et il est possible que ma méthode ne le voit pas.

CHAPITRE 5 RÉSULTATS

Lemyre Garneau (2015)

5.1 Algorithmes implémentés

noter que PB/EB est différent de PEB

5.2 Résultats numériques

métrique pour checker si sauver du temps : nb evals(temps). métrique pour checker si interruptions justifiées : objectif(temps).

in retrospecte : s3 et s7 cas qui n'ont peu ou pas les propriétés utilisés par ma méthodes. s4 cas étrange s2 bon cas.

FROM POINT NON RÉALISABLE ET PARTITION BIAISÉE

présenter dans l'ordre : s7, s3, s2, s4.
s3 : Différents cas de figure. Parfois, nomad converge rapidement sur un optimimum local qui n'est pas près d'une contrainte, donc je n'ai pas de moyen de sauver de temps, d'où les performances à peine meilleures que le cas de base, et l'idée de diviser X en sous-espaces et calculer une partition des contraintes par sous-espaces n'aidera pas. Donc s3 un peu comme s7. Parfois, c'est le cas que nomad converge près d'une contrainte et là ma méthode est gagnante. Montrer ces deux cas de figures avec seed 22 et 0 (maybe 66 ?).

pour s2 : show seed 33. Représentative des performances moyennes.

graphiques : temps solar only, nb évaluations nomad.

5.2.1 Données à priori

5.2.2 Partition optimale

5.2.3 Optimisations avec NOMAD

Comparaison des cinq algorithmes

En assumant une partition basée sur données de qualité. Aka en utilisant des données biaisées, celles qui viennent des points d'une optimisation à $\phi = 1$ de 1000 evals, en ayant réévalué les

points à chaque fidélité de Φ.

5.2.4 Comparaison des partitions

Comparaison de différentes façons d'obtenir une partitions de manière réaliste dans une implémantation réelle, donc pas avec les données biaisées de la denièere section.

méthodes comparées : tout à $\phi = 1$ (cas de base, patition sans analyse à priori), partition basée sur un HL, partition recalculée à chaque T evals.

5.3 Application potentielle à PRIAD

exemple avec priad où : phi à chaque étape et à chaque "batch" de simulations. simulations électriques c'est le plus lourd. elles sont effectuées en parallèles mais il y en a trop pour all in one shot alors il y a une queue de tâches.

certains éléments de la partition donnés par bb (modules de PRIAD) mais d'autres à déterminer avec ma méthode (ϕ de MC).

Contraintes posées sur chaque étape. Histogrammes updatés au fur et à mesure que simulations électriques faites, puis contraintes sur moyennes, variances, max, min, etc. de VoLL, de N-1, N-2, etc.

5.4 Discussion

proposition : ajouter une contrainte f < best f à date. Peut-être qu'une basse fidélité peut prédire que la valeur de f sera pas bonne.

Implémentation	**% NR**	**% R, base NR**	**% Solution moy**	**Écart-type**
Interruptions	80	0	13.24	5.07
Inter 2ph PB/PB	20	0	-11.88	7.61
Inter 2ph EB/EB	20	20	45.3	72.6
Inter 2ph PB/EB	20	0	-2.96	10.82

Tableau 5.1 Résultats de 10 optimisations de solar 2 avec différentes graines NOMAD relativement au cas de base

CHAPITRE 6 CONCLUSION

Texte.

6.1 Synthèse des travaux

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

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

Texte de l'annexe A. Remarquez que la phrase précédente se termine par une lettre majuscule suivie d'un point. On indique explicitement cette situation à LaTeX afin que ce dernier ajuste correctement l'espacement entre le point final de la phrase et le début de la phrase suivante.

ANNEXE B ENCORE UNE ANNEXE

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

ANNEXE C UNE DERNIÈRE ANNEXE

Texte de l'annexe C.