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 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.

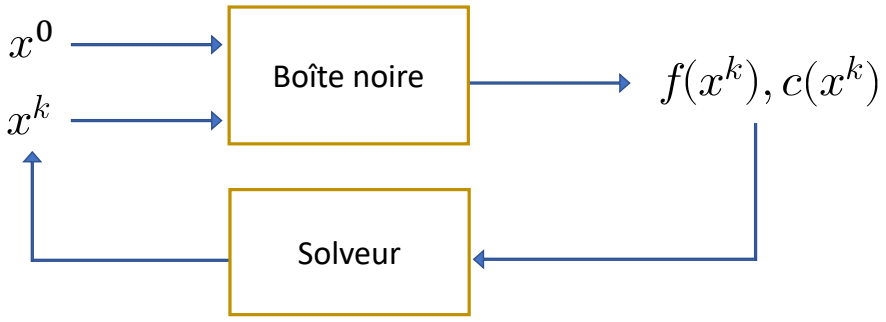


FIGURE 2.1 Généralisation des méthodes d'optimisation de boîte noire

Les algorithmes de BBO sont divisés en trois grandes catégories. La première consiste en les méthodes qui génèrent des 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.

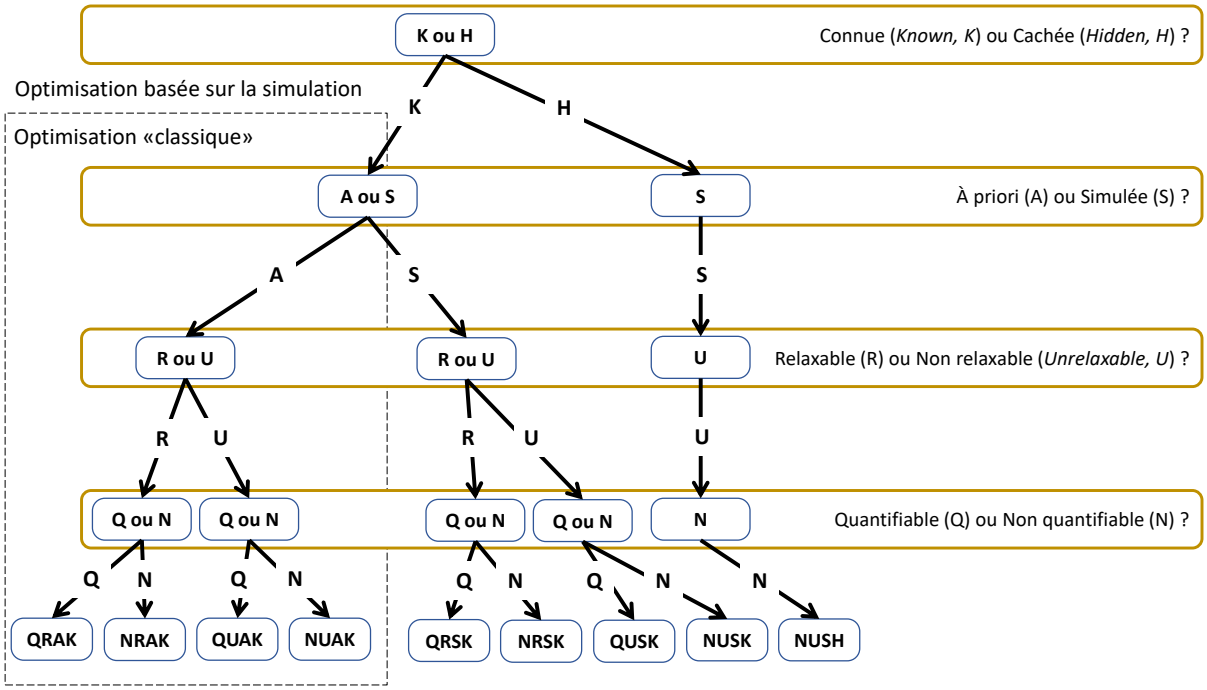


FIGURE 2.2 Représentation en arbre de la taxonomie des contraintes QRAK traduite de Le Digabel et Wild (2015)

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).

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

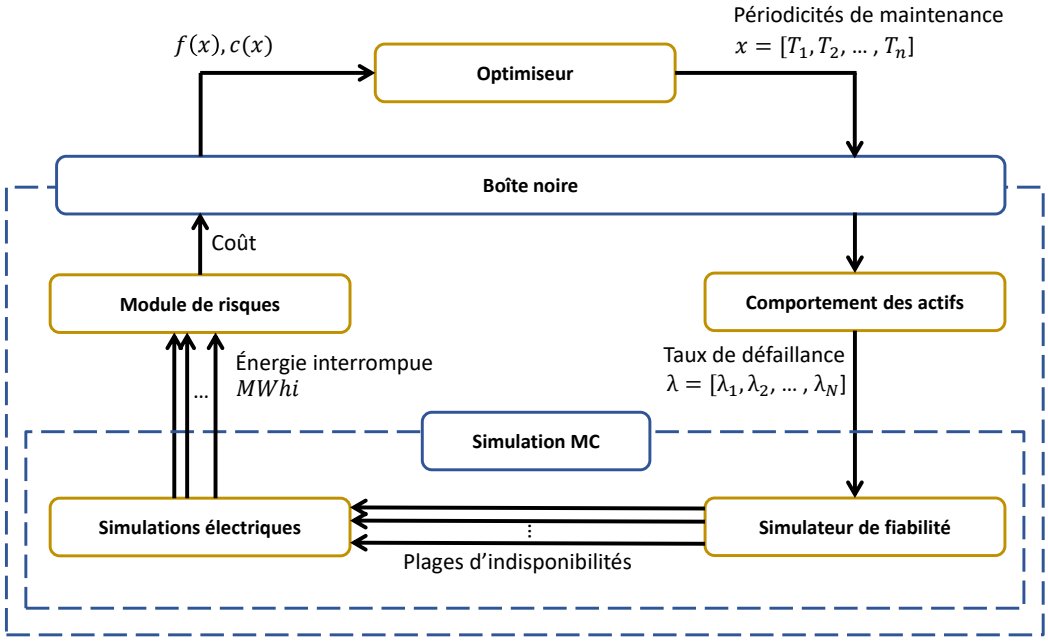


FIGURE 2.3 Représentation graphique du simulateur de PRIAD

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

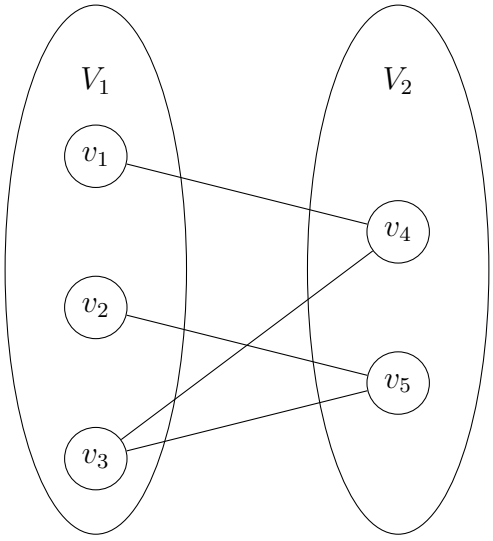
La théorie des graphes est une branche des mathématiques qui offre une manière intéressante de modéliser des problèmes d'optimisation. Il est important de mentionner qu'aucun

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.

Livre de référence à insérer ici bientôt. 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 hiérarchisée. Finalement, la dernière section décrit en détails un algorithme qui fait partie de cette méthode.

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
$\ell \in \mathbb{R}$	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 $i \in I := \{1, 2, \ldots, \ell\}$

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 la fidélité a donc été définie. Si la multi-fidélité est le résultat d'un processus stochastique, il est possible que $c(x, \phi)$ varie d'une évaluation 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 les autres définitions, 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. 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 dimension $(\ell + m) \times (\ell + m)$, alors que la matrice B contient la même information tout en étant de dimension $\ell \times m$.

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 la matrice de biadjacence optimale
3. Lancer l'optimisation avec un solveur sur une passerelle

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 l'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 fidélité 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 de l'étape trois de l'algorithme 3.1. À chaque fois que la passerelle 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.

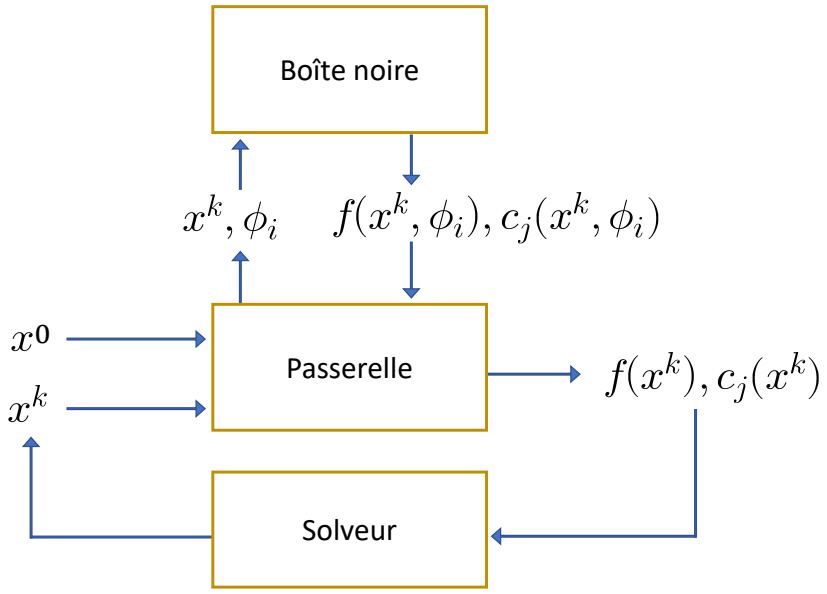


FIGURE 3.1 Boucle d'optimisation avec un solveur et la passerelle

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 point x^* dénote la meilleure solution réalisable trouvée depuis le début de l'optimisation.

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 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 termine. Autrement, la boucle principale poursuit. Si cette boucle se termine sans interruption, cela signifie que $c(x, \phi) \leq \mathbf{0}$ pour la plus grande fidélité dans Φ. Or, si cette fidélité n'est pas 1 (la vérité) et que l'objectif à la dernière sous-évaluation est meilleur

Algorithme 3.2 : Sous-évaluations interrompues

Fonction $\acute{E}valuation(x, f(x^*), Sous\text{-}évaluation(\cdot), \Phi, G)$

Déclarer f, c

Pour tout $\phi_i \in \Phi$

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

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

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

      Si $c_j > 0$

        Interruption, l'algorithme se termine

        Renvoyer f, c

Si $\max(\Phi) \neq 1$ et $f < f(x^*)$

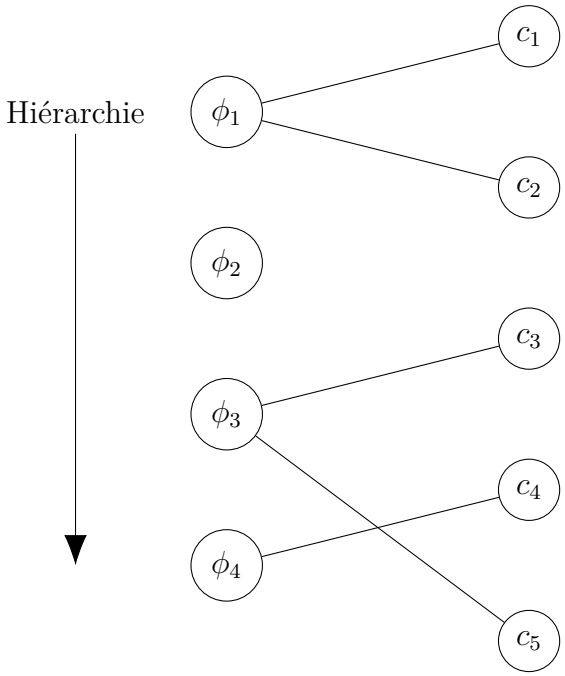
  $f, c \leftarrow Sous\text{-}évaluation(x, 1)$

Renvoyer f, c

que l'objectif au meilleur point trouvé depuis le début de l'optimisation, une sous-évaluation supplémentaire est ajoutée à la vérité. L'intérêt de cette pratique est que $f(x, \max(\Phi))$ pourrait être différent de $f(x, 1)$. La sous-évaluation ajoutée à la fin s'assure que la valeur de f renvoyée au solveur est la vrai si le solveur s'apprête à définir une nouvelle meilleure solution. De plus, il est possible que x soit réalisable à $\max(\Phi)$, mais ne le soit pas à $\phi = 1$. Cela est rare si B est de grande qualité (reflète bien le comportement des contraintes), mais il est prioritaire de garantir qu'un point qui pourrait être renvoyé à l'utilisateur si le solveur termine en tant que meilleure solution réalisable 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 s'avèrent 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.



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

FIGURE 3.2 Exemple de graphe avec sa matrice de biadjacence.

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 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 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

4.1 Représentativité des contraintes

4.2 Satisfaction des contraintes

4.3 Temps d'évaluation

4.4 Modèle d'optimisation de la matrice d'adjacence

[...]
Modèle d'optimisation analytique qui vise à trouver le meilleure fidélité à laquelle assigner chaque contrainte. La variable ε dénote probabilité estimée maximale acceptable qu'une contrainte soit mal représentée.

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

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

s.c.

$$\sum_{i \in I} B_{ij} = 1 \quad \forall\, j \in J \tag{4.1}$$

$$r_{ij} \geq (1-\varepsilon) B_{ij} \quad \forall\, i \in I, \forall\, j \in J \tag{4.2}$$

4.5 Dépendance entre la satisfaction des contraintes et la fidélité

4.6 Réductions de la taille du modèle

Définition A.1

Ensemble des fidélités minimales auxquelles la borne de représentativité ε est atteinte pour chaque contrainte.

$$\hat{I} := \bigcup_{j \in J} \min\{i \in I : r_{ij} \geq 1 - \varepsilon\} \tag{4.3}$$

Propriété A.1

Pour chaque contrainte, si la borne de représentativité ε est atteinte à une certaine fidélité, elle l'est aussi pour toute fidélité plus grande.

$$r_{aj} \geq 1-\varepsilon \implies r_{bj} \geq 1-\varepsilon \quad \forall j \in J, \forall \text{ paire } a,b \in I : b \geq a \tag{4.4}$$

Propriété A.2

Le temps est une fonction monotone croissante avec la fidélité. Cette provient de la dernière définition du tableau 2.2.

$$\phi_a < \phi_b \implies t_a \leq t_b \quad \forall \text{ paire } a,b \in I \tag{4.5}$$

Lemme A.1

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

Démonstration

La solution B^a respecte (4.2) :

$$\begin{aligned}
B^b \text{ est réalisable} &\implies r_{i'j} \geq 1-\varepsilon \quad \forall j \in J : B^b_{i'j} = 1 \\
(4.4) &\implies i' \geq \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J : B^b_{i'j} = 1 \\
i' \notin \hat{I} &\implies i' > \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J : B^b_{i'j} = 1 \\
&\implies i'-1 \geq \min\{i \in I : r_{ij} \geq 1-\varepsilon\} \quad \forall j \in J : B^b_{i'j} = 1 \\
(4.4) &\implies r_{i'-1\,j} \geq 1-\varepsilon \quad \forall j \in J : B^b_{i'j} = 1 \\
&\implies r_{i'-1\,j} \geq 1-\varepsilon \quad \forall j \in J : B^a_{i'-1\,j} = 1 \\
B^a_{ij} = B^b_{ij} \, \forall i \in I \, , \forall j \in J : B^b_{i'j} = 0 &\implies (4.2) \text{ est respectée par } B^a
\end{aligned} \tag{4.6}$$

La solution B^a respecte (4.1) :

$$\begin{aligned}
B^b \text{ est réalisable} &\implies \sum_{i\in I} B^b_{ij} = 1 \quad \forall\, j \in J \\
&\implies \sum_{i=1}^{i'-2} B^b_{ij} + 0 + 1 + \sum_{i=i'+1}^{l} B^b_{ij} = 1 \quad \forall j \in J : B^b_{i'j} = 1 \\
&\implies \sum_{i=1}^{i'-2} B^a_{ij} + 1 + 0 + \sum_{i=i'+1}^{l} B^a_{ij} = 1 \quad \forall j \in J : B^a_{i'-1\,j} = 1 \\
&\implies \sum_{i\in I} B^a_{ij} = 1 \quad \forall\, j \in J \\
&\implies (4.1) \text{ est respectée par } B^a \qquad (4.7)
\end{aligned}$$

$$(4.6) \text{ et } (4.7) \implies B^a \text{ est réalisable}$$

□

Lemme A.2

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

Démonstration

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

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

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

Deux scénarios peuvent survenir, dépendamment de la valeur de $y^b_{i'-1}$.
Si $y^b_{i'-1} = 0$:

$$\begin{aligned}
f(B^b) =& t_1 y^b_1 + \sum_{i=2}^{i'-2} t_i M^b_i + t_{i'-1} M^b_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i M^b_l \\
=& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^b_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i M^a_l \\
& \qquad \text{car } B^b_{ij} = B^a_{ij},\ \forall i \in I \backslash \{i'-1, i'\},\ \forall j \in J \\
=& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^b_{i'-1} + t_{i'} M^a_{i'-1} + \sum_{i=i'+1}^{l} t_i M^a_l \\
& \qquad \text{car } M^b_{i'} = M^a_{i'-1} \\
\geq& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^b_{i'-1} + t_{i'-1} M^a_{i'-1} + \sum_{i=i'+1}^{l} t_i M^a_l \\
& \qquad \text{car } t_{i'} \geq t_{i'-1} \text{ car 4.5} \\
=& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^a_{i'-1} + t_{i'} M^a_{i'} + \sum_{i=i'+1}^{l} t_i M^a_l \\
& \qquad \text{car } t_{i'} M^a_{i'} = t_{i'-1} M^b_{i'-1} = 0 \\
=& f(B^a)
\end{aligned} \tag{4.8}$$

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

$$\begin{aligned}
f(B^b) =& t_1 y^b_1 + \sum_{i=2}^{i'-2} t_i M^b_i + t_{i'-1} M^b_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i M^b_l \\
=& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'-1} M^b_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i M^a_l \\
& \qquad \text{car } B^b_{ij} = B^a_{ij},\ \forall i \in I \backslash \{i'-1, i'\},\ \forall j \in J \\
=& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i y^a_i \overset{i-1^a}{\underset{k=1}{M}} + t_{i'-1} M^a_{i'-1} + t_{i'} M^b_{i'} + \sum_{i=i'+1}^{l} t_i y^a_i \overset{l-1^a}{\underset{k=1}{M}} \\
& \qquad \text{car } t_{i'-1} M^b_{i'-1} = t_{i'-1} M^a_{i'-1} \\
>& t_1 y^a_1 + \sum_{i=2}^{i'-2} t_i M^a_i + t_{i'} M^a_{i'} + t_{i'-1} M^a_{i'-1} + \sum_{i=i'+1}^{l} t_i M^a_l \\
& \qquad \text{car } t_{i'} M^b_{i'} > 0 = t_{i'} M^a_{i'} \\
=& f(B^a)
\end{aligned} \tag{4.9}$$

$$(4.8) \text{ et } (4.9) \implies f(B^a) \leq f(B^b)$$

□

Théorème 1

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

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

□