

Titre: Optimisation de boîtes noires multifidélités avec contraintes
Title: hiérarchisées

Auteur: Xavier Lebeuf
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Lebeuf, X. (2023). Optimisation de boîtes noires multifidélités avec contraintes
Citation: hiérarchisées [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie.
<https://publications.polymtl.ca/53386/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/53386/>
PolyPublie URL:

Directeurs de recherche: Charles Audet, Sébastien Le Digabel, & Miguel Diago-Martinez
Advisors:

Programme: Maitrise recherche en mathématiques appliquées
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Optimisation de boîtes noires multifidélités
avec contraintes hiérarchisées**

XAVIER LEBEUF

Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Avril 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Optimisation de boîtes noires multifidélités
avec contraintes hiérarchisées**

présenté par **Xavier LEBEUF**

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

Hanane DAGDOUGUI, présidente

Charles AUDET, membre et directeur de recherche

Sébastien LE DIGABEL, membre et codirecteur de recherche

Miguel DIAGO-MARTÍNEZ, membre et codirecteur de recherche

Antoine LESAGE-LANDRY, membre

DÉDICACE

Cet univers désormais sans maître ne lui paraît ni stérile, ni fertile. Chacun des grains de cette pierre, chaque éclat minéral de cette montagne pleine de nuit, à lui seul, forme un monde. La lutte elle-même vers les sommets suffit à remplir un cœur d'homme. Il faut imaginer Sisyphe heureux.

Albert Camus

REMERCIEMENTS

En premier lieu, je remercie mes directeurs de l'école Polytechnique Montréal Charles Audet et Sébastien Le Digabel, pour m'avoir accueilli dans leur équipe et pour m'avoir conseillé judicieusement tout au long de ce projet de recherche. Ils ont su nourrir ma passion avec leur expertise théorique ainsi que leur connaissance du milieu académique. Aussi, je remercie Karine Hébert, toute l'équipe ainsi que les autres étudiants du GÉRAD pour avoir créé un environnement agréable et favorable au développement de mon projet.

En second lieu, j'exprime ma gratitude envers mon co-directeur Miguel Diago-Martínez ainsi que Stéphane Alarie, tous deux chercheurs de l'Institut de recherche en électricité du Québec (IREQ). Ils ont su me faire parvenir une vision concrète des enjeux réels qui touchent mon projet, de manière à mieux guider celui-ci. Je remercie également Hussein Suprême ainsi que l'équipe Contrôle et gestion des réseaux de l'IREQ pour m'avoir intégré dans leur unité et pour m'avoir accordé du support lorsque besoin fut.

Aussi, j'éprouve une vive reconnaissance à l'endroit de Sara Séguin, professeure à l'Université du Québec à Chicoutimi, qui m'a introduit à ce fascinant domaine d'étude il y a plusieurs années, et sans qui mon parcours académique n'aurait probablement pas été autant comblé d'expériences stimulantes et enrichissantes.

Finalement, je remercie tendrement ma famille et mes amis qui ont su me supporter moralement, souvent sans même le savoir.

RÉSUMÉ

On propose un algorithme d'optimisation de boîtes noires multifidélités qui s'intéresse au cas où une grande proportion du temps d'optimisation utilisé par les algorithmes de recherche directe est dépensé sur des points non réalisables. La méthode proposée doit être couplée avec un solveur existant, et elle permet à celui-ci de réduire le temps espéré des évaluations en estimant, à l'aide d'évaluations peu coûteuses, si un point est réalisable avant d'y investir plus de temps. Ces estimations sont obtenues avec une hiérarchisation des contraintes affectées par la multifidélité, qui est définie par une matrice de biadjacance. On propose une méthode de calcul de cette matrice.

La recherche présentée s'inscrit dans le projet Alliance du Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) auquel participent Polytechnique Montréal et Hydro-Québec. À l'IREQ, un projet de recherche de stratégie de maintenance optimale requiert l'optimisation d'une boîte noire particulièrement coûteuse en temps qui contient plusieurs contraintes rarement satisfaites. L'algorithme est développé avec l'intention d'être appliqué à ce problème lorsque le projet d'Hydro-Québec atteindra cette phase.

Dans le cadre de ce projet, des tests numériques sont effectués avec la famille de boîtes noires **solar**. Le solveur Optimisation non linéaire par recherche directe sur treillis adaptatifs : *Nonlinear Optimisation by Mesh Adaptive Direct search* (NOMAD) couplé à l'algorithme de hiérarchisation des contraintes est comparé au solveur NOMAD avec ses paramètres par défaut. Ces tests révèlent que l'algorithme permet de trouver des solutions significativement meilleures lorsqu'un point de départ réalisable est connu avant l'optimisation. Sans cette condition, les résultats sont variables; ils dépendent grandement des propriétés de la boîte noire optimisée.

ABSTRACT

We propose a multi-fidelity blackbox optimization algorithm that addresses the problem of having to spend large computational resources on infeasible points when using direct search algorithms. The proposed method is coupled with an existing solver, allowing a decrease of the expected time per evaluation while keeping the efficiency and convergence properties of the existing method. This is achieved by estimating the feasibility of points to evaluate with low fidelity, hence low cost, evaluations before deciding if a point is worth the full time investment. These estimations are given by a hierarchy of constraints that are affected by the multi-fidelity, which is defined by a biadjacency matrix. We propose a computation method for this matrix.

The project is part of the Natural Sciences and Engineering Research Council of Canada (NSERC) Alliance program. At Hydro-Quebec’s research institute : *Institut de recherche en électricité du Québec* (IREQ), an optimal maintenance strategy problem requires the optimization of a particularly costly blackbox, containing many rarely satisfied constraints. The proposed algorithm is developed with the intention of being applied to the optimal maintenance strategy problem when the project reaches the optimization stage.

During this project, numerical tests are conducted on the **solar** family of blackbox problems. The Nonlinear Optimisation by Mesh Adaptive Direct search (NOMAD) software is used as the existing solver, and the solver with default parameters is compared to the solver coupled with the proposed algorithm during the tests. These tests show that given the same time budget, the coupling with the proposed method results in a great improvement in the quality of the solutions when a feasible starting point is known prior to the optimization. Without this condition, the results are mixed and largely depend on some properties of the optimized blackbox.

TABLE DES MATIÈRES

DÉDICACE	iii
REMERCIEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE DES MATIÈRES	vii
LISTE DES TABLEAUX	x
LISTE DES FIGURES	xi
LISTE DES ALGORITHMES	xiii
LISTE DES EXEMPLES	xiv
LISTE DES SIGLES ET ABRÉVIATIONS	xv
LISTE DES ANNEXES	xvi
CHAPITRE 1 INTRODUCTION	1
1.1 Contexte	1
1.2 Objectifs de recherche	2
1.3 Plan du mémoire	3
CHAPITRE 2 REVUE DE LITTÉRATURE	5
2.1 Optimisation de boîtes noires	5
2.2 Gestion des contraintes	9
2.3 Multifidélité	12
2.4 Boîtes noires stochastiques	15
2.5 PRIAD	15
2.6 Réduction du temps d'optimisation	18
2.7 Théorie des graphes	19
CHAPITRE 3 SOUS-ÉVALUATIONS INTERROMPUES	21

3.1	Notation et définitions	21
3.2	Algorithme d'optimisation avec contraintes hiérarchisées	22
3.3	Algorithme de sous-évaluations interrompues	24
3.4	Application potentielle à PRIAD	27
CHAPITRE 4 MATRICE DE BIADJACENCE ET OPTIMISATION		32
4.1	Analyse du comportement des contraintes	32
4.2	Modèle d'optimisation de la matrice de biadjacence	34
4.3	Différences entre le modèle et le vrai problème	36
4.4	Résolution rapide du modèle	38
4.4.1	Réduction 1	38
4.4.2	Réduction 2	39
4.4.3	Réduction 3	45
4.4.4	Réduction 4 et résolution	45
4.5	Algorithme d'optimisation avec contraintes hiérarchisées détaillé	45
CHAPITRE 5 BANC D'ESSAI ET RÉSULTATS INTERMÉDIAIRES		48
5.1	Banc d'essai et implémentations	48
5.2	Analyse du comportement des contraintes	50
5.2.1	solar2	51
5.2.2	solar3	53
5.2.3	solar4	57
5.2.4	solar7	59
5.2.5	Conclusion sur les analyses du comportement des contraintes	62
5.3	Calcul de la matrice de biadjacence	62
5.3.1	solar2	62
5.3.2	solar3	64
5.3.3	solar4	65
5.3.4	solar7	65
5.4	Conclusion sur les résultats intermédiaires	67
CHAPITRE 6 RÉSULTATS DES OPTIMISATIONS		69
6.1	Validation du calcul de la matrice de biadjacence	70
6.1.1	solar2	71
6.1.2	solar3	71
6.1.3	solar4	72
6.2	Résultats numériques des optimisations	73

6.2.1	Avec un point de départ non réalisable provenant d'un échantillon . .	74
6.2.2	Avec un point de départ réalisable provenant d'un échantillon	77
6.2.3	Avec un point de départ réalisable connu	79
6.3	Discussion	83
CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS		86
7.1	Synthèse des travaux et limitations	86
7.2	Améliorations futures	87
RÉFÉRENCES		89
ANNEXES		95

LISTE DES TABLEAUX

Tableau 2.1	Définitions liées à l'optimisation de boîtes noires.	6
Tableau 2.2	Définitions liées à la multifidélité.	14
Tableau 2.3	Temps d'évaluation du simulateur de PRIAD.	18
Tableau 3.1	Définitions liées à l'algorithme de sous-évaluations interrompues.	21
Tableau 5.1	Caractéristiques de quatre boîtes noires solar multifidélités. .	48
Tableau 5.2	Réalisabilité d'échantillons de 10^4 points pour 4 instances solar .	49
Tableau 5.3	Configurations pour l'analyse du comportement des contraintes.	50
Tableau 5.4	Valeur de c_2 à trois points de l'hypercube latin centré de solar4 .	55
Tableau 5.5	Réalisabilité à priori des configurations de solar4	57
Tableau 6.1	Réalisabilité d'optimisations de solar2 avec différents $x^0 \notin \Omega$.	75
Tableau 6.2	Réalisabilité d'optimisations de solar3 avec différents $x^0 \notin \Omega$.	77
Tableau 6.3	Réalisabilité d'optimisations de solar4 avec différents $x^0 \notin \Omega$.	77
Tableau B.1	Bornes des hypercubes latins avec $\rho = 1$	98
Tableau B.2	Bornes des hypercubes latins avec $\rho < 1$	100

LISTE DES FIGURES

Figure 2.1	Généralisation des méthodes d'optimisation de boîte noire. . .	7
Figure 2.2	Représentation en arbre de la taxonomie des contraintes QRAK. . .	13
Figure 2.3	Représentation graphique du simulateur de PRIAD.	16
Figure 3.1	Boucle d'optimisation avec l'algorithme 3.2.	25
Figure 3.2	Application potentielle à PRIAD	29
Figure 5.1	Valeurs des r_{ij} de solar2 avec différents hypercubes latins. . . .	51
Figure 5.2	Valeurs des p_{ij} de solar2 avec différents hypercubes latins. . . .	52
Figure 5.3	Valeurs des t_i de solar2 avec différents hypercubes latins. . . .	53
Figure 5.4	Valeurs des r_{ij} de solar3 avec différents hypercubes latins. . . .	54
Figure 5.5	Valeurs des p_{ij} de solar3 avec différents hypercubes latins. . . .	55
Figure 5.6	Valeurs des t_i de solar3 avec différents hypercubes latins. . . .	57
Figure 5.7	Valeurs des r_{ij} de solar4 avec différents hypercubes latins. . . .	58
Figure 5.8	Valeurs des p_{ij} de solar4 avec différents hypercubes latins. . . .	59
Figure 5.9	Valeurs des t_i de solar4 avec différents hypercubes latins. . . .	60
Figure 5.10	Valeurs des r_{ij} de solar7 avec différents hypercubes latins. . . .	60
Figure 5.11	Valeurs des p_{ij} de solar7 avec différents hypercubes latins. . . .	61
Figure 5.12	Valeurs des t_i de solar7 avec différents hypercubes latins. . . .	61
Figure 5.13	Graphes réduits représentant les assignations de solar2	63
Figure 5.14	Graphes réduits représentant les assignations de solar3	64
Figure 5.15	Graphes réduits représentant les assignations de solar4	66
Figure 5.16	Graphes réduits représentant les assignations de solar7	67
Figure 6.1	Optimisations de solar2 avec différentes stratégies de calcul de B . . .	72
Figure 6.2	Optimisations de solar3 avec différentes stratégies de calcul de B . . .	72
Figure 6.3	Optimisations de solar4 avec différentes stratégies de calcul de B . . .	73
Figure 6.4	Optimisation de solar2 à partir du meilleur point non réalisable. . .	74
Figure 6.5	Optimisation de solar2 à partir du 5e point non réalisable. . . .	74
Figure 6.6	Optimisation de solar2 à partir du 14e point non réalisable. . . .	75
Figure 6.7	Profils de données de solar2 à partir de points non réalisables. . .	76
Figure 6.8	Optimisation de solar2 à partir d'un point de départ réalisable. . .	78
Figure 6.9	Profils de données de solar2 à partir de points réalisables. . . .	78
Figure 6.10	Optimisation de solar7 à partir d'un point de départ réalisable. . .	79
Figure 6.11	Profils de données de solar7 à partir de points réalisables. . . .	79
Figure 6.12	Optimisation de solar2 à partir d'un point réalisable connu. . . .	80

Figure 6.13	Profils de données de solar2 avec un hypercube latin centré. . .	81
Figure 6.14	Optimisation de solar3 à partir d'un point réalisable connu. . .	81
Figure 6.15	Profils de données de solar3 avec un hypercube latin centré. . .	82
Figure 6.16	Optimisation de solar4 à partir d'un point réalisable connu. . .	82
Figure 6.17	Profils de données de solar4 avec un hypercube latin centré. . .	83
Figure A.1	Graphes réduits donnés par des stratégies alternatives pour solar2 .	95
Figure A.2	Graphes réduits donnés par des stratégies alternatives pour solar3 .	96
Figure A.3	Graphes réduits donnés par des stratégies alternatives pour solar4 .	97

LISTE DES ALGORITHMES

2.1	Barrière extrême à deux phases, tiré de Audet et Hare (2017)	10
2.2	Mise à jour des points x^{fea} et x^{inf} et du seuil h de la BP	11
3.1	Optimisation avec contraintes hiérarchisées	24
3.2	Sous-évaluations interrompues	26
4.1	Optimisation avec contraintes hiérarchisées (détaillé)	47

LISTE DES EXEMPLES

2.1	Graphe biparti avec ses matrices d'adjacence et de biadjacence.	20
3.1	Assignations illustrées par un graphe et sa matrice de biadjacence.	23
3.2	Intervalles de temps dans PRIAD.	30
4.1	Problème de matrice de biadjacence optimale simple.	40
4.2	Contre-exemple à l'hypothèse (4.11).	43

LISTE DES SIGLES ET ABRÉVIATIONS

CRSNG	Conseil de recherches en sciences naturelles et en génie du Canada
NSERC	Natural Sciences and Engineering Research Council of Canada
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 (<i>Derivative Free Optimization</i>)
BBO	Optimisation de boîtes noires (<i>Blackbox Optimization</i>)
NOMAD	Optimisation non linéaire par recherche directe sur treillis adaptatifs (<i>Nonlinear Optimisation by Mesh Adaptive Direct search</i>)
BP	Barrière progressive
BE	Barrière extrême
MADS	Recherche directe sur treillis adaptatifs (<i>Mesh Adaptive Direct Search</i>)
MC	Monte-Carlo
BPE	Barrière progressive à extrême

LISTE DES ANNEXES

Annexe A	Graphes réduits de différentes stratégies de calcul de B	95
Annexe B	Bornes des hypercubes latins	98

CHAPITRE 1 INTRODUCTION

La recherche effectuée dans ce mémoire est motivée par un problème rencontré par l'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. L'institut souhaite optimiser la stratégie de maintenance des équipements électriques du réseau de transport d'Hydro-Québec. L'un des objectifs de PRIAD est d'élaborer un simulateur du réseau électrique 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, pour associer un coût à cet ensemble d'intervalles, 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 donné par le simulateur. L'intuition derrière ce second objectif est qu'effectuer toutes les maintenances sur le réseau quotidiennement serait trop coûteux en ressources monétaires et humaines, et inversement, trop peu de maintenance causerait une trop grande quantité de défaillances sur le réseau. Le but est d'optimiser les périodicités en trouvant celles qui font le meilleur compromis, c'est-à-dire celles qui minimisent le coût total.

Cet ouvrage propose un nouvel algorithme d'optimisation qui exploite l'aspect multifidélité d'une boîte noire pour trouver de meilleures solutions avec un budget de temps donné. Cet algorithme est développé avec l'intention d'être appliqué au problème de PRIAD ainsi qu'à tout problème d'optimisation donné par une boîte noire multifidélité contraint. L'algorithme effectue des appels séquentiels à la boîte noire en imposant différentes fidélités selon la satisfaction des diverses contraintes. Un nouveau problème émerge de cet algorithme, qui est de déterminer une séquence d'appels et de vérifications des contraintes efficace. La génération de cette séquence nécessite une matrice de biadjacence, et une méthode de construction de cette matrice est proposée.

1.1 Contexte

De par sa nature informatique, il n'est pas possible d'extraire un ensemble d'équations analytiques formant un modèle mathématique qui définit le comportement du simulateur de PRIAD. 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 mé-

thodes d’optimisation sans dérivées : *Derivative Free Optimization* (DFO) et d’optimisation de boîtes noires : *Blackbox 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. À titre d’exemple, une contrainte peut imposer que le nombre d’évènements où plusieurs équipements sont simultanément indisponibles doit être inférieur ou égal à une certaine quantité lors des simulations électriques. 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 d’optimisation 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 une grande quantité 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 un grand nombre de tirages, le temps total d’optimisation devient très grand. Dans le cas de PRIAD, sans aucune méthode de réduction du temps de calcul (comme le parallélisme par exemple), 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.

1.2 Objectifs de recherche

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 multifidélité 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 multifidélité ne peut bénéficier de cette méthode.

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

Un second objectif est de tester la méthode développée avec plusieurs implémentations différentes. La méthode d'optimisation avec contraintes hiérarchisée comprend un algorithme développé pour agir en tant que passerelle entre un algorithme d'optimisation et la boîte noire à optimiser. De ce fait, la méthode doit être couplée avec un 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 gérer la précision variable de la boîte noire indépendamment de cet algorithme. Dans ce projet, la méthode est testée avec l'algorithme Recherche directe sur treillis adaptatifs : *Mesh Adaptive Direct Search* (MADS), et avec différentes méthodes de gestion des contraintes. Le logiciel NOMAD est utilisé pour les tests.

1.3 Plan du mémoire

La présentation de ce mémoire est divisée en sept chapitres. Le second présente une revue de la littérature qui aborde plusieurs sujets pertinents. Le chapitre 3 présente la méthode proposée d'un point de vue haut niveau, et il détaille ensuite un algorithme utilisé par la méthode. Aussi, il discute de l'application de la méthode au problème du projet PRIAD. Le chapitre 4 s'intéresse à l'aspect mathématique du problème, et se termine avec une présentation détaillée de la méthode proposée. Le chapitre 5 présente des résultats numériques provenant d'étapes intermédiaires de la méthode d'optimisation avec contraintes hiérarchisées. Le chapitre 6 présente les résultats numériques de l'application complète de la méthode, où des optimisations sont effectuées et comparées. Ce chapitre se termine avec une discussion

sur l'ensemble des résultats. Enfin, le chapitre 7 comporte la conclusion de cet ouvrage, qui présente une synthèse des travaux, qui discute des limitations de la méthode proposée et qui suggère des améliorations futures.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce second chapitre vise à établir les notions prérequis 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 multifidélité, 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 est peu mentionné pour les raisons mentionnées dans l'introduction. Il est tout de même important de couvrir le sujet dans cette revue de littérature, car 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. À titre d'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 multiobjectifs ne sont pas considérés.

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

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

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
$f : X \rightarrow \mathbb{R} \cup \{\infty\}$	Fonction qui renvoie la valeur de l'objectif au point x
$c : X \rightarrow \mathbb{R}^m \cup \{\infty\}^m$	Fonction qui renvoie les valeurs des contraintes relaxables au point x
$c_j(x) \leq 0$	j -ième contrainte relaxable, où $j \in J := \{1, 2, \dots, m\}$

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.

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

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

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

de l'information. 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.

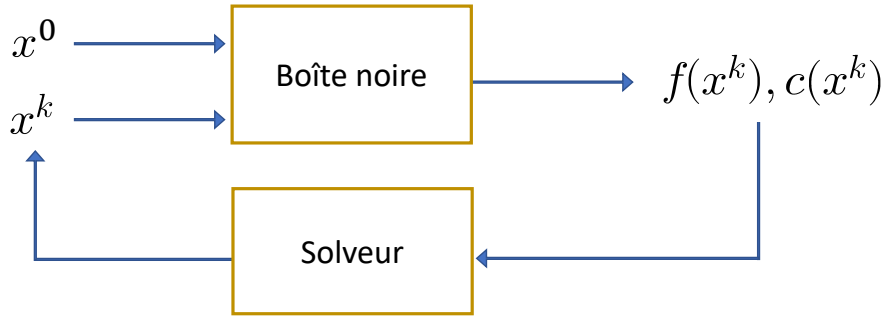


FIGURE 2.1 Généralisation des méthodes d'optimisation de boîte noire.

Les algorithmes de BBO sont divisés en trois grandes catégories. La première comprend 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 substituts entiers-mixtes : *Mixed-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 $n + 1$ arêtes). 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 dépendants du simplexe de départ. Huyer et Neumaier (1999) ont proposé l'algorithme Recherche par coordonnées multiniveaux : *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 de BBO est vaste, et la quantité d'implémentations logicielles différentes de ces algorithmes l'est tout aussi. 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. Un 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 ne domine tous les autres. Plus récemment, une analyse similaire a été effectuée par Ploshkas 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. Entre autres, 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 coûteux 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 $h : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ 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 $x \in X \setminus \Omega$. 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 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 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 au long de l'optimisation. La BP introduit un seuil $h_{\max} \in \mathbb{R}^+$, mis à jour à chaque itération de manière qu'il soit monotone décroissant selon les itérations de l'algorithme d'optimisation. Chaque point x où $h(x) > h_{\max}$ est ignoré en attribuant à $f(x)$ une valeur de ∞ . Le seuil initial est donné par $h_{\max} = \infty$.

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

Étant donné $f_\Omega : X \rightarrow \mathbb{R} \cup \{\infty\}$, $c : X \rightarrow \mathbb{R}^m \cup \{\infty\}^m$ 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 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 ∞ . Ces deux points ainsi que $h_{\max}$ sont mis à jour à l'aide de la notion de dominance défini ci-bas.

Le point $x \in \Omega$ domine le point $y \in \Omega \iff f(x) < f(y)$,

dénoté par $x \prec_f y$.

Le point $x \in X \setminus \Omega$ domine le point $y \in X \setminus \Omega \iff$

$f(x) \leq f(y)$ et $h(x) \leq h(y)$ et au moins une de ces inégalités est stricte,

dénoté par $x \prec_h y$.

l'Algorithme 2.2 détaille la mise à jour des points x^{fea} et x^{inf} et du seuil $h_{\max}$, effectuée à la fin de chaque itération de l'algorithme d'optimisation. L'ensemble V est défini comme l'ensemble de tous les points pour lesquels l'algorithme a calculé la valeur de h et de f du début de l'optimisation à la plus récente itération.

Algorithme 2.2 : Mise à jour des points x^{fea} et x^{inf} et du seuil h de la BP

Étant donné $h_{\max}$, x^{fea} , x^{inf} et le dernier point évalué x

Si $x \in \Omega$ avec $x \prec_f x^{\text{fea}}$

$h_{\max} \leftarrow h(x^{\text{inf}})$ $x^{\text{fea}} \leftarrow x$
--

Sinon $x \in X \setminus \Omega$ avec $x \prec_h x^{\text{inf}}$

$h_{\max} \leftarrow h(x^{\text{inf}})$ $x^{\text{inf}} \leftarrow x$
--

Sinon $x \in X \setminus \Omega$ avec $0 < h(x) < h(x^{\text{inf}})$

$h_{\max} \leftarrow \max\{h(v) : h(v) < h(x^{\text{inf}}), v \in V\}$ $x^{\text{inf}} \leftarrow \operatorname{argmin}_v\{f(v) : h(v) \leq h_{\max}, v \in V\}$

Sinon

$h_{\max} \leftarrow h(x^{\text{inf}})$

L'algorithme d'optimisation couplé avec la BP explore alors autour de x^{fea} ainsi qu'autour de x^{inf} . L'algorithme 2.2 ne détaille pas cette exploration car elle dépend de l'algorithme d'optimisation utilisé. L'intérêt est que x^{inf} est souvent un point avec une meilleure valeur de f que x^{fea} , et en faisant progresser la valeur de $h_{\max}$ vers 0, il est possible de trouver un point réalisable près de x^{inf} très intéressant.

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 (BPE). 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. (2023) 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’optimalité similaire 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 par un arbre une série de questions à 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.

2.3 Multifidélité

Cette section décrit le concept de multifidélité 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 de qualité dégradée, alors qu’une haute fidélité correspond à des sorties fiables et coûteuses. 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 multifidélité s’il peut être effectué à différents niveaux de fidélité. Une grande branche de la BBO s’intéresse à la bifidélité, un cas particulier de la multifidélité 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 multifidélité, la méthode présentée dans ce mémoire s’applique aussi à

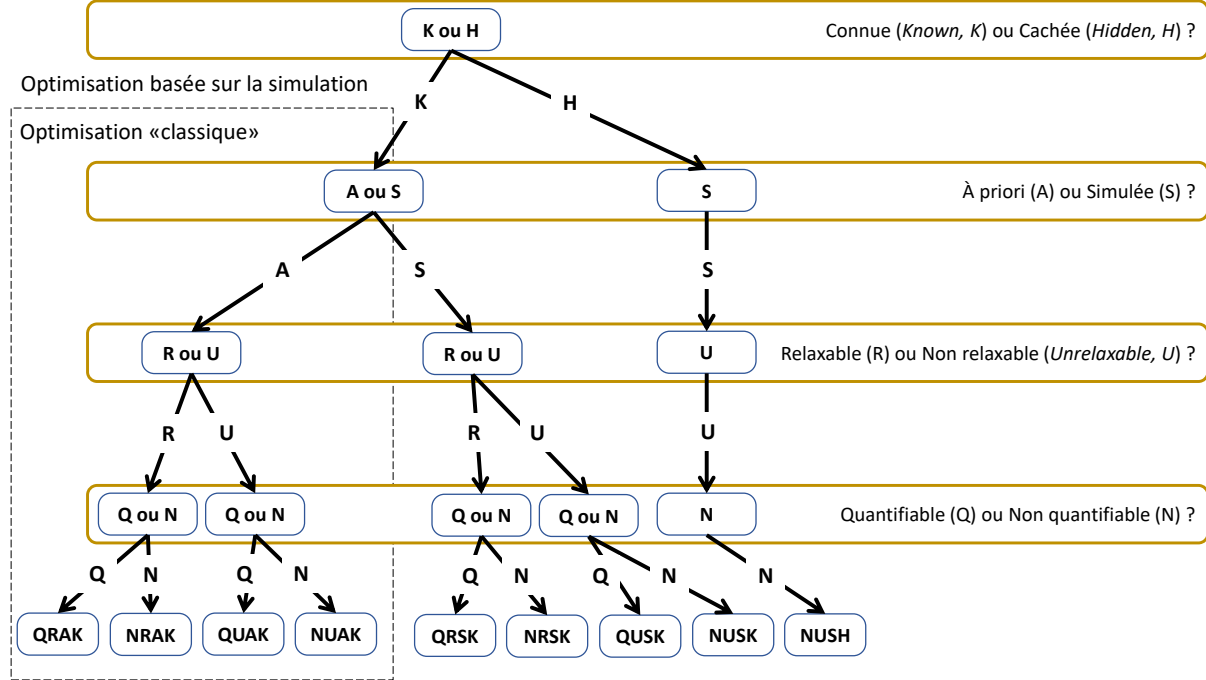


FIGURE 2.2 Représentation en arbre de la taxonomie des contraintes QRAK traduite de Le Digabel et Wild (2015).

la bifidélité, mais les méthodes spécialisées en bifidélité risquent d'être plus efficaces lorsque seulement deux fidélités sont disponibles. Un exemple de telle méthode est celle proposée par Balabanov et Venter (2004), où la basse-fidélité est utilisée pour évaluer des gradients.

Pour citer un exemple de procédé multifidélité 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 ainsi que 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 multifidélité, basées sur celles présentées par Sen et al. (2018). La notation est adaptée au contexte de ce mémoire.

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 MC 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

TABLEAU 2.2 Définitions liées à la multifidélité.

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

méthode qui ajuste la précision d'une boîte noire sans contraintes au fur et à mesure de l'optimisation qui s'applique aux simulations MC. À la seconde définition du tableau 2.2, la définition de f du tableau 2.1 est étendue en ajoutant la fidélité comme paramètre. La notation $f(x)$ implique alors que $\phi = 1$. La troisième définition du tableau 2.2 impose que λ^c augmente avec la fidélité. Cette définition sera utile pour garantir l'optimalité d'un sous-problème d'optimisation détaillé à la section 4, où une diminution de la fidélité doit impliquer une diminution du temps d'évaluation. La variable λ^c est définie comme le coût à des fins de généralité. Or, pour ce projet de maîtrise, ce coût correspond au temps d'évaluation.

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

Pour lister quelques nouveaux exemples, une boîte noire multifidélité 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. Ce dernier cas est étudié 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 multifidélités comprend une grande variété de problèmes. Bien sûr, toute boîte noire multifidélité 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 multiobjectif multifidélité. Aussi, Wang et al. (2022) proposent une

extension à l’algorithme d’arbre de recherche optimiste hiérarchique : *hierarchical optimistic tree search* pour utiliser des évaluations à basse fidélité.

2.4 Boîtes noires stochastiques

La multifidélité est souvent le produit d’un processus stochastique. En revanche, les méthodes d’optimisation de boîtes noires stochastiques s’appliquent différemment des méthodes d’optimisation de boîtes noires multifidélités. 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 multifidélité, la précision est contrôlable par un ou plusieurs paramètres mappés sur la fidélité qui appartient à $[0, 1]$. Les approches sont donc plutôt distinctes.

Ce travail de maîtrise teste la méthode proposée avec une implémentation de l’algorithme MADS, pour lequel des adaptations pour les boîtes noires stochastiques ont été développées récemment. Entre autres, 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’estimation 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 similaire 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.

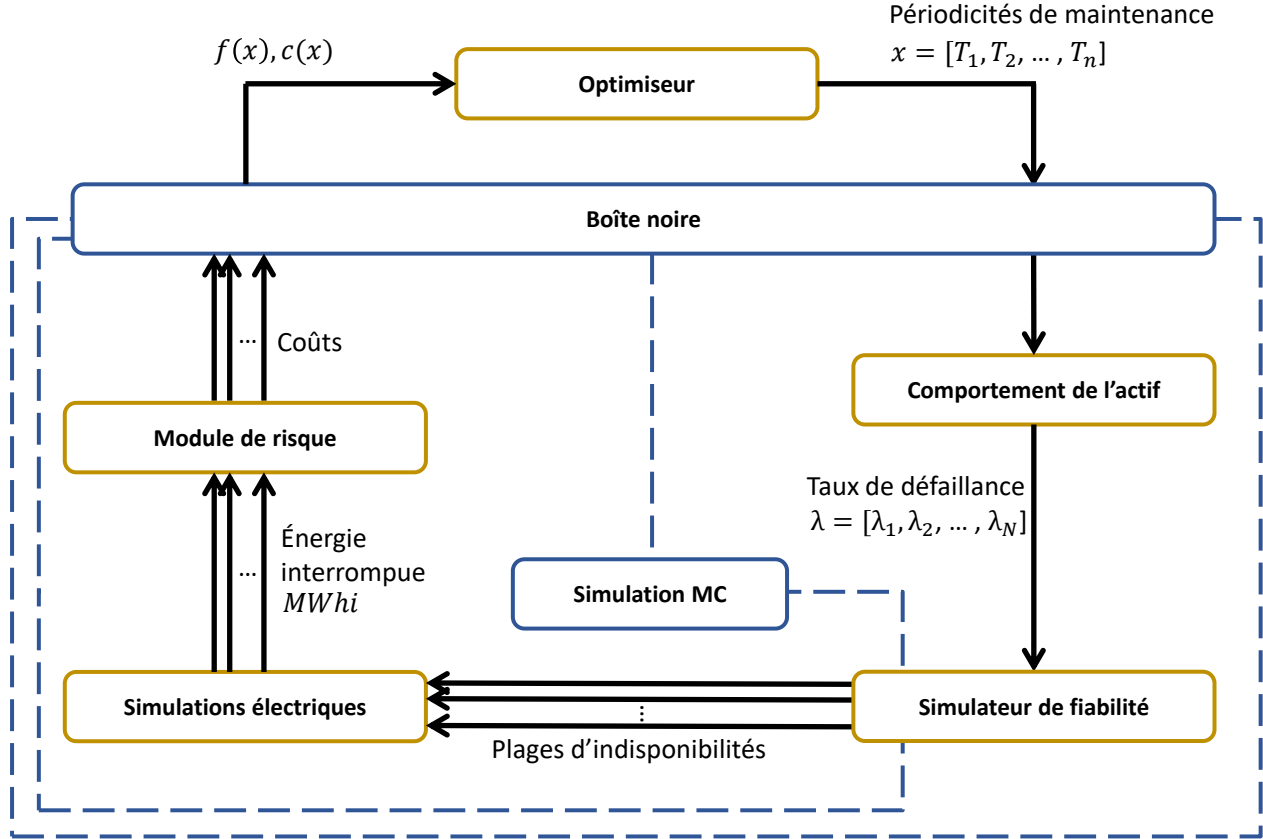


FIGURE 2.3 Représentation graphique du simulateur de PRIAD.

D’abord, l’optimiseur envoie un point de taille n à la boîte noire. Ce dernier correspond aux périodicités de maintenance T 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, les lignes de transport, 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 défaillance. Ce modèle est alors comparé à des données historiques pour être calibré. 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 des simulations de fiabilité. 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’horizon de temps est divisé en périodes selon les combinaisons d’équipements indisponibles en même temps. Pour chaque combinaison rencontrée (pour chaque période), les mécanismes de protection du réseau sont simulés, et il en résulte une configuration du réseau donnée par les états « ouvert » ou « fermé » des disjoncteurs et des sectionneurs. Chaque configuration unique à travers tous les tirages MC est appelée un scénario. Par la suite, l’avant-dernier module effectue une simulation de l’écoulement de puissance pour chacun de ces scénarios. Durant chaque simulation, les limites principales suivantes sont surveillées :

- limite thermique des équipements ;
- limite de tension aux barres du réseau ;
- limite de perte de charge et de production.

Lorsqu’une limite est violée, les actions d’un exploitant sont également simulées. Les actions principales disponibles sont les suivantes :

- reprogrammation de la production ;
- reconfiguration du réseau ;
- rappel d’équipements au retrait (remettre en fonction un équipement usé) ;
- coupure des échanges avec les réseaux voisins ;
- délestage.

Chaque simulation renvoie l’énergie non livrée sur le réseau, indiquée par l’acronyme *MWhi* qui signifie mégawattheure interrompus. Finalement, le module de risque traduit les sorties des différents modules en termes monétaires. Plus précisément, il calcule une valeur de la charge perdue : *value of lost load* (VoLL). 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. À titre d’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 (le nombre d’équipements électriques sur le réseau de transport d’Hydro-Québec est supérieur à 2000).

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

Cas	Nombre d'équipements	Nombre de tirages MC	Nombre de scénarios 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	3.5×10^9	145

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 :

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

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 coupé sur les évaluations des points qui ne sont pas meilleurs que le meilleur point courant. Le second algorithme repose sur une hiérarchie des sous-boîtes noires. D'abord, les contraintes sont classées de la plus rarement satisfaite à la plus fréquemment satisfaite. 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 qui risquent le plus de causer une interruption. 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 modélisation intuitive pour les problèmes d'optimisation où elle est applicable. 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.

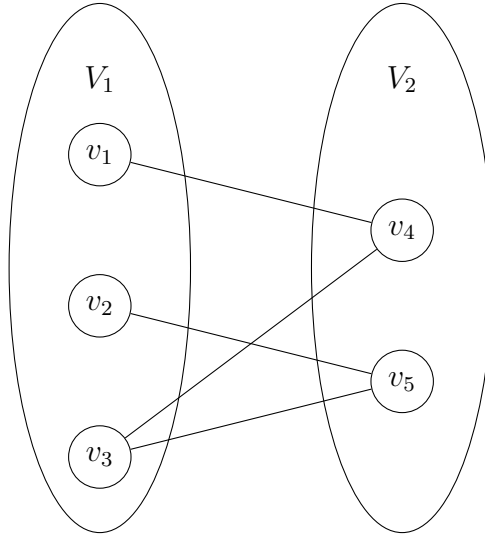
Un graphe G est défini par deux ensembles. Un premier ensemble V contenant les sommets, et un second ensemble E contenant les arêtes qui relient les sommets entre eux. Une

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 ensemble 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 de l'information redondante, et peut être écrite sous 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. L'exemple 2.1 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.

Exemple 2.1 Graphe biparti avec ses matrices d'adjacence et de biadjacence.



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

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

CHAPITRE 3 SOUS-ÉVALUATIONS INTERROMPUES

Ce chapitre vise à montrer comment la méthode proposée réduit le temps d'optimisation d'une boîte noire contrainte multifidélité. La première section de ce chapitre établit certaines définitions qui sont utilisées dans le reste de ce mémoire. La seconde section décrit la méthode proposée, appelée la méthode d'optimisation avec contraintes hiérarchisées. La troisième section décrit en détail un algorithme qui fait partie de la méthode d'optimisation avec contraintes hiérarchisées. Finalement, dans une dernière section, il est décrit comment la méthode pourrait potentiellement être implémentée avec le problème de PRIAD.

3.1 Notation et définitions

De nouvelles définitions sont données au tableau 3.1.

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

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

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

ϕ . Si la multifidélité est le résultat d'un processus stochastique, il est possible que $c(x, \phi)$ et $t(x, \phi)$ varient d'une évaluation à l'autre pour le même x et le même ϕ . Dans ce cas, $c(\cdot)$ et $t(\cdot)$ ne sont pas des fonctions. En revanche, pour une graine aléatoire fixe, $c(\cdot)$ et $t(\cdot)$ sont des fonctions. Ce constat est également valide pour $f(x, \phi)$. Concernant les deux dernières définitions du tableau 3.1, sachant que $\phi \in [0, 1]$, il existe une quantité infinie de fidélités auxquelles il est possible d'appeler une boîte noire. Comme la méthode présentée dans ce mémoire ne peut que considérer une quantité finie de fidélités, l'utilisateur doit choisir un ensemble Φ de taille $\ell \in \mathbb{N}$. Cet ensemble doit contenir la fidélité 1 pour le déroulement de l'analyse. Le choix de la taille de Φ dépend de la capacité de calcul parallèle de l'utilisateur. Cet aspect est détaillé dans un prochain chapitre.

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

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

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 des arêtes E est donné par la matrice de biadjacence B . Puisque le graphe G est biparti, la matrice d'adjacence de G n'est pas utilisée. En effet, celle-ci est de taille $(\ell + m) \times (\ell + m)$, alors que la matrice B contient la même information tout en étant de taille $\ell \times m$. Pour illustrer ces propos, l'exemple 3.1 montre les assignations d'un problème où $m = 5$ et $\ell = 4$ avec un graphe, ainsi que la matrice de biadjacence du graphe. On y remarque que l'indice

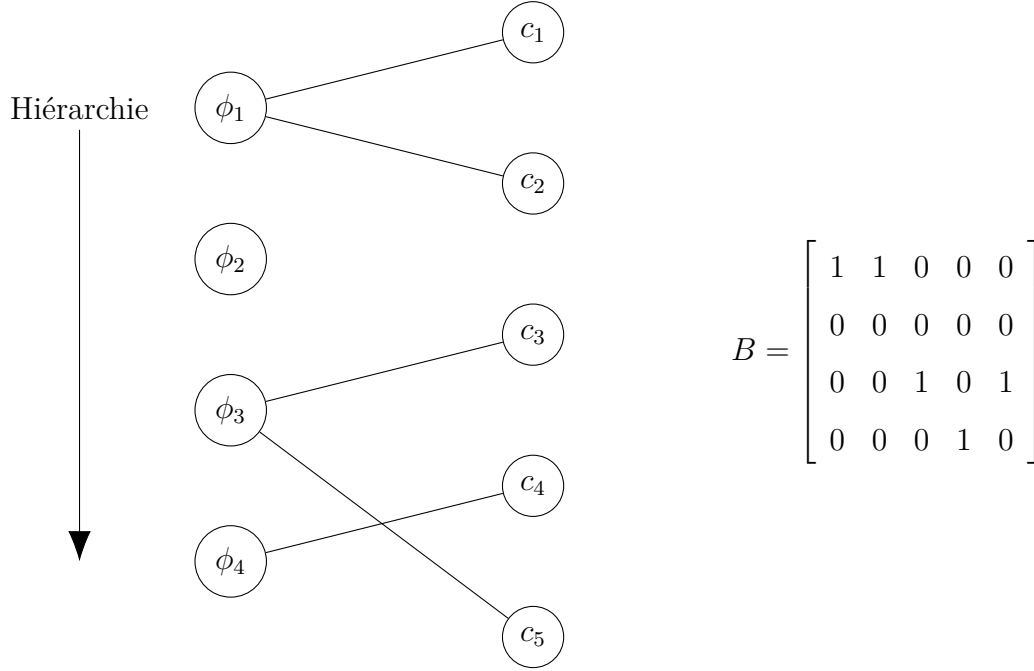
$$B_{ij} = \begin{cases} 1 & \text{si la contrainte } c_j \text{ est assignée à la fidélité } \phi_i \\ 0 & \text{sinon.} \end{cases}$$

Le calcul de cette matrice est étudié au chapitre 4.

3.2 Algorithme d'optimisation avec contraintes hiérarchisées

La méthode proposée consiste à assigner les contraintes d'un problème d'optimisation de boîte noire à des fidélités. L'objectif de ces assignations est de refléter à partir de quelle

Exemple 3.1 Assignations illustrées par un graphe et sa matrice de biadjacence.



fidélité chaque contrainte donne de l'information pertinente, où $\phi = 1$ est la vérité, donc l'information la plus pertinente. À l'exemple 3.1, en supposant que les assignations sont justes, le graphe indique qu'une évaluation à la fidélité ϕ_1 suffit pour déterminer si les deux premières contraintes du problème sont satisfaites. Des appels à la boîte noire aux fidélités de degré non nul dans G peuvent permettre de déterminer si certaines contraintes ne sont pas satisfaites, et ce, rapidement si ces fidélités sont basses. L'algorithme se termine dès qu'il est identifié qu'une contrainte n'est pas satisfaite. On dit alors qu'il y a une interruption. 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.

Il est nécessaire que l'ensemble des arêtes E soit déterminé en fonction du comportement des contraintes à différentes fidélités pour qu'il permette des interruptions justes. Si, à l'exemple 3.1, une évaluation à ϕ_1 révèle que la contrainte c_2 n'est pas satisfaite alors qu'elle serait respectée si une évaluation à $\phi = 1$ était effectuée, l'assignation de la contrainte c_2 est mauvaise, et une interruption injuste a lieu. De ce fait, la première étape consiste à effec-

Algorithme 3.1 : Optimisation avec contraintes hiérarchisées

Étant donné un problème d'optimisation donné par une boîte noire multifidélité

1. Analyser le comportement des contraintes en fonction de la fidélité
 2. Calcul d'une matrice de biadjacence optimale
 3. Optimisation de la boîte noire
-

tuer une analyse qui identifie, entre autres, la pertinence de l'information donnée par chaque contrainte à chaque fidélité de Φ . La seconde étape se base sur cette analyse pour calculer une matrice de biadjacence optimale. Sachant que le comportement de la méthode durant l'optimisation dépend des assignations des contraintes, cette étape vise à trouver la matrice de biadjacence B qui minimise l'espérance du temps d'évaluation. Après l'étape deux, le graphe G est défini à partir de V et de E . Finalement, à la dernière étape, l'optimisation est lancée. Durant cette optimisation, chaque évaluation est constituée de plusieurs appels à différentes fidélités. Après chaque appel, s'il est estimé que le point en cours d'évaluation n'est pas réalisable, l'algorithme se termine. Si l'analyse de la première étape reflète adéquatement à quelles fidélités chaque contrainte donne des valeurs pertinentes, il y a la possibilité d'effectuer des interruptions lors d'évaluations de points non réalisables sans avoir à faire des appels coûteux à la boîte noire.

3.3 Algorithme de sous-évaluations interrompues

Cette section détaille la troisième étape de l'algorithme 3.1. Durant l'optimisation, un solveur d'optimisation de boîtes noires existant est lancé. Toutefois, au lieu de donner au solveur la boîte noire directement, l'algorithme de sous-évaluations interrompues est donné pour gérer l'aspect multifidélité et effectuer les interruptions. Cette relation est illustrée à la figure 3.1. Cette figure reprend le schéma de la figure 2.1 pour visualiser l'insertion de l'algorithme. Chaque fois que le solveur soumet un point pour son évaluation (et avec x^0), l'algorithme de sous-évaluations interrompues appelle la boîte noire séquentiellement à chaque fidélité à laquelle est assignée au moins une contrainte (chaque fidélité de degré non nul dans G), dans l'ordre donné par 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. Du moment que l'une de ces contraintes n'est pas respectée, le point en évaluation est estimé non réalisable et l'algorithme est interrompu, ce qui met fin à l'évaluation. Les lancements effectués par l'algorithme 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.

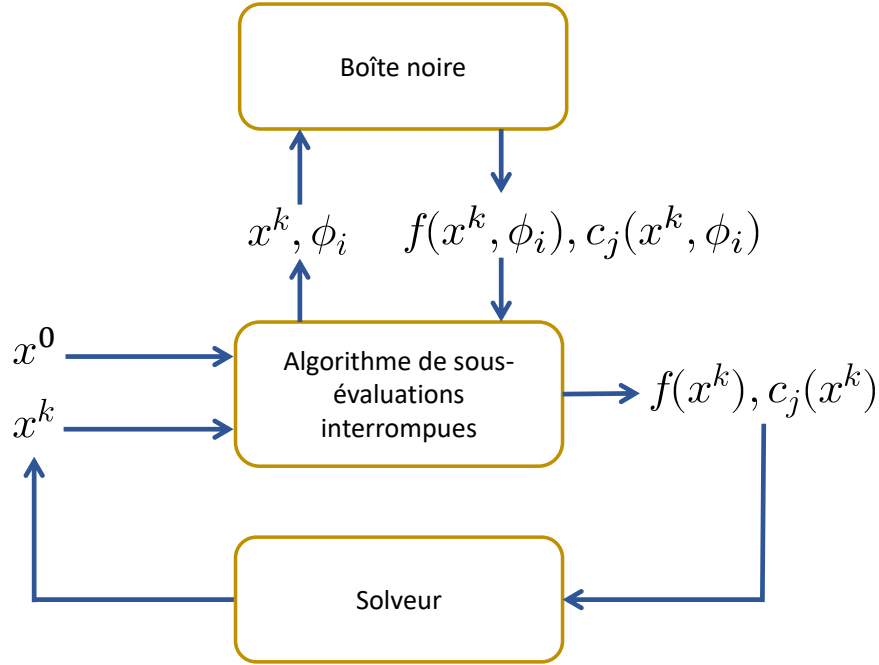


FIGURE 3.1 Boucle d'optimisation avec l'algorithme de sous-évaluations interrompues.

Cette méthode est détaillée 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. 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 nul. La fonction *Sous-évaluation* : $(\mathbb{R}^n, [0, 1]) \rightarrow (\mathbb{R} \cup \{\infty\}, (\mathbb{R} \cup \{\infty\})^m)$ dénote un lancement de la boîte noire au point x avec la fidélité ϕ , et elle renvoie $f(x, \phi)$ et $c(x, \phi)$. La variable f^* dénote la valeur de la fonction objectif à la meilleure solution réalisable trouvée depuis le début de l'optimisation. Cette valeur est obtenue en lançant une fonction nommée *Obtenir-meilleure-valeur*. Cette dernière pourrait, par exemple, lire un fichier pour renvoyer f^* . Avant d'avoir trouvé un premier point réalisable, la fonction renvoie ∞ .

En suivant la hiérarchie, l'algorithme lance la boîte au point x à toute fidélité $\phi_i \in \Phi$ si $e_i^\top B \neq \mathbf{0}$. Cette condition assure de ne pas effectuer de sous-évaluation si aucune contrainte n'est assignée à ϕ_i . Après chaque exécution de la boîte noire, une boucle itère sur les j qui sont tels que $1 \in \{B_{aj} \in B : a \leq i\}$. Cet ensemble contient les éléments de la matrice

Algorithme 3.2 : Sous-évaluations interrompues

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

 Déclarer f, c
 $f^* \leftarrow \text{Obtenir-meilleure-valeur}()$
Pour tout $\phi_i \in \Phi$

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

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

 | **Pour tout** $j \in J$ tel que $1 \in \{B_{aj} \in B : a \leq i\}$

 | | Si $c_j > 0$

| | | Interruption, l'algorithme se termine

 | | **Renvoyer** f, c

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

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

B correspondant à la contrainte c_j et à toute fidélité égale ou inférieure à ϕ_i . Si l'un de ces éléments vaut 1, la contrainte est vérifiée, et une interruption survient si elle n'est pas satisfaite. Sachant que la pertinence de l'information ne fait qu'augmenter avec la fidélité, il est justifié de vérifier chaque contrainte à la fidélité à laquelle elle est assignée ainsi qu'à chaque fidélité supérieure où une sous-évaluation est effectuée. De plus, il n'y a aucun coût significatif associé à simplement vérifier une contrainte. À l'exemple 3.1, cela signifie qu'après l'appel à ϕ_3 , en plus de c_3 et c_5 , les contraintes c_1 et c_2 sont revérifiées. Aussi, à la dernière sous-évaluation, toutes les contraintes sont vérifiées. Si la boucle itérant sur les ϕ_i se termine sans interruption, cela signifie que $c(x, \phi) \leq \mathbf{0}$ pour la plus grande fidélité dans Φ à laquelle est assignée au moins une contrainte. Or, malgré que $1 \in \Phi$, il est possible que cette fidélité ne soit pas 1. Il est possible que la matrice B optimale déterminée avant l'optimisation soit telle que $e_\ell^\top B = \mathbf{0}$ (avec $\phi_\ell = 1$). De plus, si la valeur de la fonction objectif à la dernière sous-évaluation est meilleure que pour le meilleur point trouvé depuis le début de l'optimisation, une sous-évaluation supplémentaire est ajoutée à la fidélité correspondant à la vérité ($\phi = 1$). L'intérêt de cette pratique est qu'il est possible que x soit réalisable avant la sous-évaluation supplémentaire, mais ne le soit pas à $\phi = 1$. Cela est rare si B est de qualité (reflète bien le comportement des contraintes), mais il est prioritaire de garantir qu'un point qui pourrait être renvoyé à l'utilisateur en tant que meilleure solution réalisable si le solveur se termine

soit véritablement réalisable.

Le temps d'une évaluation dépend principalement des interruptions. Si, dans l'exemple 3.1, les contraintes c_1 et c_2 sont rarement satisfaites à ϕ_1 , les évaluations constitueront souvent un seul un appel à la boîte noire à la fidélité la moins coûteuse. En revanche, si aucune interruption n'a lieu, le temps d'une évaluation est au mieux la somme des temps des sous-évaluations à ϕ_1 , ϕ_3 et ϕ_4 . Cette observation met en valeur la pertinence d'une matrice B qui minimise l'espérance du temps. Chaque cas où plusieurs sous-évaluations sont effectuées doit être compensé par un ou plusieurs cas (dépendamment des temps d'évaluation en fonction de la fidélité) où des interruptions surviennent. Sinon, ne pas utiliser l'algorithme d'optimisation avec contraintes hiérarchisées et simplement fixer la fidélité à 1 est plus rapide. 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.

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 fidélités inférieures à 1 ;
- l'existence d'optimums 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 entraîneront 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.

3.4 Application potentielle à PRIAD

Cette section s'intéresse à comment la méthode s'applique à PRIAD. Il est à noter que le projet de l'IREQ est en développement, et que cette section présente une application potentielle de la méthode d'optimisation avec contraintes hiérarchisées.

Une sous-évaluation a été définie précédemment comme le lancement d'une boîte noire par

l'algorithme de sous-évaluations interrompues. 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. En particulier, dans le cas de PRIAD, la simulation peut renvoyer des sorties après l'exécution du premier module, après l'exécution du second module et après certaines quantités de tirages MC. Chaque fidélité correspond à un moment durant la simulation où des sorties sont disponibles. Dans ce contexte, à chaque évaluation, l'algorithme 3.2 doit être adapté pour ne lancer la boîte noire qu'une seule fois, et recevoir 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 . Il est à noter que les boîtes noires avec cette propriété sont peu fréquentes.

La fidélité ϕ_1 correspond à l'achèvement du module de comportement de l'actif. Ce module renvoie des taux de défaillance λ , sur lesquels il est possible de poser qu'aucun λ ne doit excéder un seuil $\bar{\lambda}$. Il s'agit d'une contrainte qui est toujours assignée à ϕ_1 , indépendamment de l'analyse du comportement des contraintes et du calcul de B . La fidélité ϕ_2 correspond à l'achèvement du simulateur de fiabilité. Ce module calcule une plage de disponibilité par tirage MC. Après cela, il est possible de déterminer, pour chaque plage, le nombre d'évènements où un équipement est indisponible $N - 1$, le nombre d'évènements où deux équipements sont indisponibles $N - 2$, et ainsi de suite. Pour chacun de ces nombres d'indisponibilités concourantes, il est possible de formuler un histogramme, et de poser des contraintes sur la moyenne, l'écart-type, la médiane ou différents centiles. Ces contraintes sont toujours assignées à ϕ_2 , indépendamment des autres assignations. Ces deux premières fidélités correspondent à des temps négligeables en comparaison aux simulations électriques. Même si ces simulations électriques sont lancées en parallèle, elles sont trop lourdes et trop nombreuses; elles sont la cause du grand temps d'évaluation de la boîte noire. C'est donc le calcul de ces simulations qui est décomposé en fidélités. Le module de risque prend également un temps négligeable, ce qui permet de l'appeler après chaque tirage MC pour y attribuer un coût. À titre d'exemple, chaque fidélité pourrait correspondre à une tranche de e simulations. À l'atteinte de chaque fidélité, un histogramme des MWh_i et un histogramme des coûts sont formulés à partir de toutes les simulations effectuées depuis le début de l'évaluation. Encore une fois, il est possible de poser des contraintes sur la moyenne, l'écart-type, la médiane ou différents centiles, et ce, sur les MWh_i ainsi que sur les coûts. La méthode d'optimisation

avec contraintes hiérarchisée présentée dans ce mémoire devra alors déterminer à partir de quelle fidélité (quel nombre de tirages MC) chacune de ces contraintes peut être assignée. Il peut aussi y avoir des contraintes sur des maximums. Par exemple, si le coût du tirage le plus coûteux doit être inférieur à $\overline{cout}$. Comme il est désirable d'arrêter l'évaluation dès qu'un seul tirage excède un maximum, ces contraintes sont nécessairement assignées à ϕ_3 . La figure 3.2 illustre ces propos.

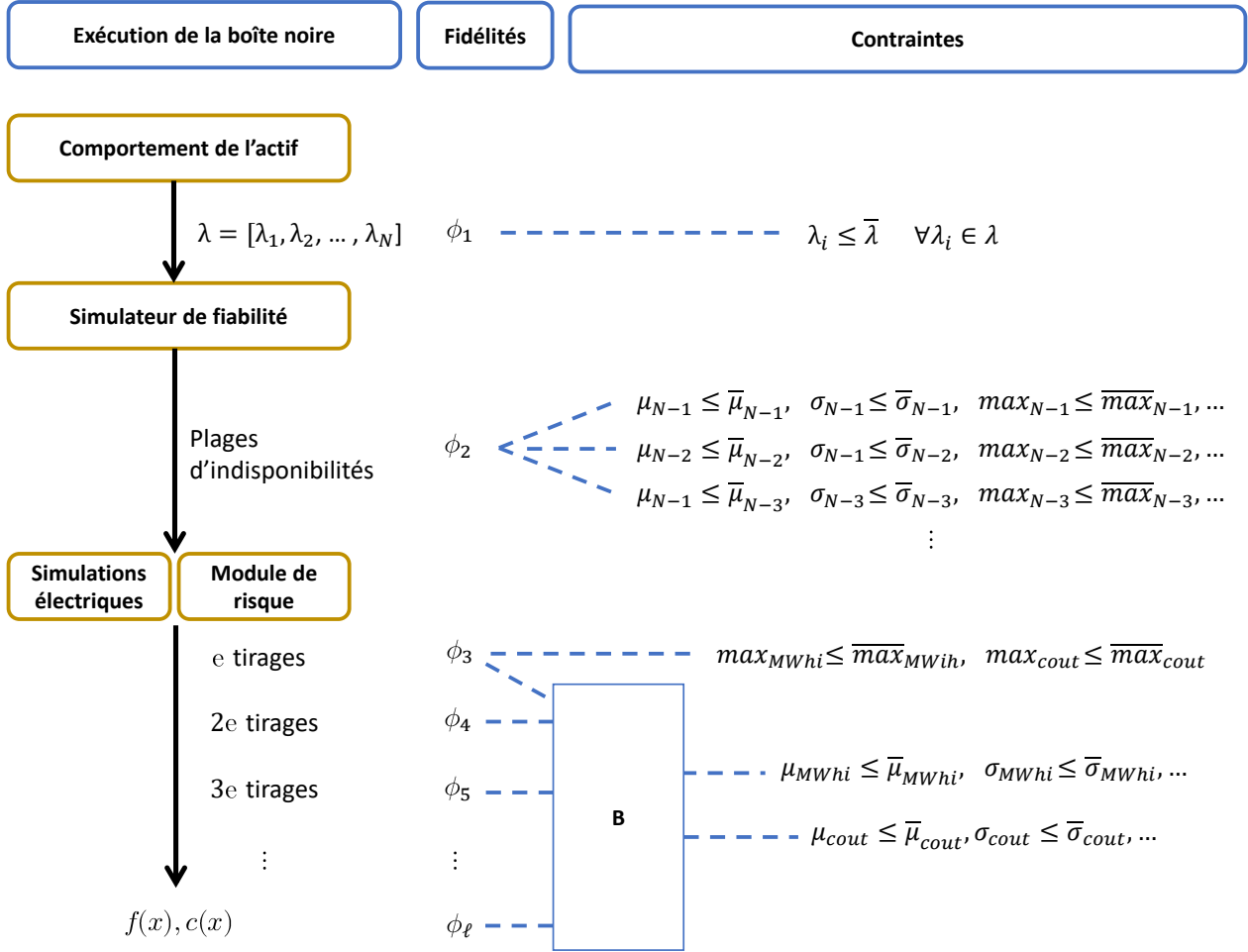


FIGURE 3.2 Application potentielle à PRIAD

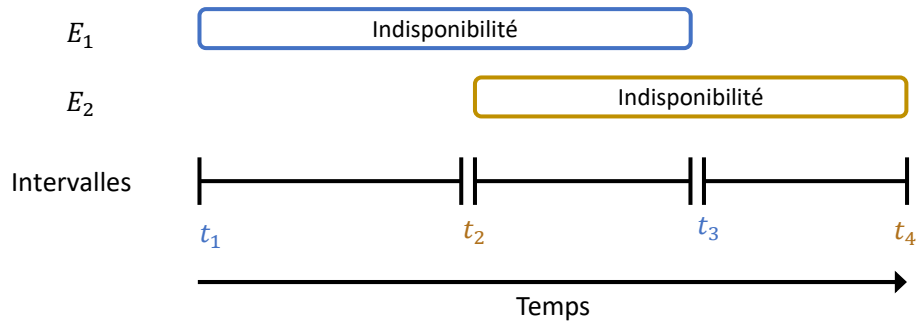
Dans cette figure, la colonne de gauche exprime les étapes importantes de l'exécution de la boîte noire. Les tirages MC sont divisés en tranches de e , ce qui est utile lorsque les simulations électriques sont parallélisées en lots de e tâches. L'écart en nombre de tirages entre les fidélités peut également être donné par une fonction polynomiale, exponentielle, ou autre. La colonne du centre associe une fidélité à chaque étape. Dans la colonne de droite, des exemples de contraintes qu'il est possible de poser sont donnés. La notation μ_a signifie la moyenne du paramètre a de l'échantillon, σ_a exprime l'écart-type, $\max_a$ exprime la plus

grande valeur de a dans l'échantillon, et les points de suspension représentent les autres contraintes que l'on peut poser sur différents centiles de a , sur des probabilités que a atteigne certaines valeurs, et ainsi de suite. Une variable avec une barre au-dessus représente un seuil maximal. Les valeurs de ces seuils proviennent d'experts du domaine, et de la tolérance des opérateurs du réseau. À titre d'exemple, si un opérateur n'est pas confortable avec un écart-type sur les $MWhi$ supérieur à $\overline{\sigma_{MWhi}}$, on pose la contrainte $\sigma_{MWhi} \leq \overline{\sigma_{MWhi}}$. En pratique, même si la moyenne des mégawatts interrompus pour une solution est acceptable, un écart-type trop grand signifie que par moments il n'y aura aucune défaillance alors qu'à d'autres moments, il y aura des pannes graves, ce qui n'est pas désirable. Finalement, les traits bleus représentent des assignations. Certaines de celles-ci sont fixes, et d'autres sont déterminées par une matrice de biadjacence B .

Des conditions peuvent également être posées sur les périodicités T elles-mêmes. Si ces conditions sont sous forme de bornes, elles peuvent être vues comme des contraintes non relaxables qui définissent l'espace des paramètres X .

Pour terminer, le projet PRIAD offre un second levier pour réduire le temps des évaluations. Détaillons d'abord ce qui constitue les simulations électriques par rapport aux tirages MC. Pour chaque plage d'indisponibilité (chacune correspondant à un tirage MC) générée par le simulateur de fiabilité, l'horizon de temps est découpé en un nombre minimal d'intervalles où chaque intervalle ne doit pas comprendre de début ou d'arrêt d'indisponibilité d'un équipement. L'exemple 3.2 présente un cas simple d'horizon de temps d'une plage d'indisponibilités d'un réseau de deux équipements. Durant cette section, l'équipement E_1 est indisponible du temps t_1 au temps t_3 , et l'équipement E_2 est indisponible de t_2 à t_4 . L'horizon est alors découpé en trois intervalles : un premier où seulement E_1 est indisponible, un second où E_1 et E_2 sont indisponibles, et un dernier où seulement E_2 est indisponible.

Exemple 3.2 Intervalles de temps dans PRIAD.



Pour chaque intervalle, les mécanismes de protection du réseau sont simulés étant donné les indisponibilités, et il en résulte une configuration du réseau donnée par les états « ouvert » ou « fermé » des disjoncteurs et des sectionneurs. Chaque configuration unique est appelée un scénario. Un scénario peut survenir plusieurs fois à travers le même tirage, ou à travers tous les tirages. Une simulation électrique est lancée par scénario, et un tirage est terminé quand tous les scénarios qui y sont compris ont été simulés. En plus de la fidélité, il est possible de déterminer l'ordre dans lequel sont effectuées les simulations électriques pour réduire le temps des évaluations de la boîte noire. Sachant qu'une interruption peut être effectuée à chaque fidélité, il est possible de prioriser les scénarios qui surviennent dans la plus grande proportion des tirages. En terminant les tirages plus rapidement, les interruptions surviendront plus rapidement. Aussi, il est possible de prioriser les scénarios par leur importance. À titre d'exemple, si un scénario qui est constitué de l'indisponibilité de plusieurs transformateurs importants qui sont proches dans le réseau risque d'être la cause de contraintes non respectées. Effectuer ce scénario en premier peut accélérer une interruption. Finalement, la pertinence de définir les fidélités par des nombres de simulations électriques (nombre de scénarios) plutôt que des nombres de tirages MC devra être étudiée.

CHAPITRE 4 MATRICE DE BIADJACENCE ET OPTIMISATION

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

4.1 Analyse du comportement des contraintes

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

D'abord, les bornes de l'hypercube latin doivent être déterminées. Il est possible d'échantillonner l'entièreté de l'espace des paramètres si les bornes de l'hypercube latin sont définies par les vecteurs $l \in \mathbb{R}^n$ et $u \in \mathbb{R}^n$ (les bornes de X). Dans le contexte de ce projet de maîtrise, on souhaite obtenir de l'information sur les points qui risquent d'être évalués lors de l'optimisation. Il peut alors être préférable d'évaluer un hypercube latin centré sur le point de départ de l'optimisation, et avec des bornes plus rapprochées du point de départ. On définit un nouveau paramètre $\rho \in [0, 1]$, le facteur de rapprochement des bornes. Les bornes sont également centrées sur le point de départ x^0 . Si une borne est à l'extérieur de X après l'avoir centré, elle est redéfinie par la borne de X . Ces nouvelles bornes sont nommées l^{cen} et u^{cen} ,

et elles sont calculées avec

$$\begin{aligned} l_i^{\text{cen}} &= \max \left(l_i, x_i^0 - \rho(u_i - l_i) \right) \quad \forall i \in \{1, 2, \dots, n\} \\ u_i^{\text{cen}} &= \min \left(u_i, x_i^0 + \rho(u_i - l_i) \right) \quad \forall i \in \{1, 2, \dots, n\}. \end{aligned}$$

Avec $\rho = 1$, les bornes sont celles de X , peu importe où se situe le point de départ. Avec $\rho = 0$, les bornes définissent un ensemble singleton qui ne contient que x^0 .

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

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

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

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

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

Autrement dit, pour un certain x et une certaine c_j , les fidélités représentatives sont celles supérieures à la plus grande fidélité n'identifiant pas correctement si la contrainte est satisfaite ou non. C'est la raison pourquoi Φ est défini comme un ensemble auquel appartient

1. L'analyse consiste à déterminer les estimations suivantes pour un $x \in X$ quelconque :

- $r_{ij} : =$ Estimation de la probabilité que la fidélité $\phi_i \in \Phi$
soit représentative pour la contrainte c_j .
- $p_{ij} : =$ Estimation de la probabilité que c_j soit satisfaite à $\phi_i \in \Phi$
 $=$ Estimation de $Pr[\beta(c_j(x, \phi_i)) = 0]$.
- $t_i : =$ Estimation du temps moyen d'une sous-évaluation à $\phi_i \in \Phi$.

Soit H l'ensemble contenant tous les points de l'hypercube latin, et n_H le nombre de points de cet hypercube. Définissons également H^{ap} , un sous-ensemble de H qui contient les points réalisables à priori (qui satisfont les contraintes de catégorie « à priori » selon la taxonomie QRAK de la figure 2.2), ainsi que n_H^{ap} la taille de cet ensemble. Les estimations sont obtenues en calculant des proportions ou des moyennes sur les points de H^{ap} . Plus la taille de l'échantillon n_H^{ap} est grande, plus ces proportions et moyennes sont près des vraies probabilités et des vraies moyennes, respectivement. La probabilité estimée inverse de p_{ij} , nommée $\bar{p}_{ij}$ est également calculée :

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

L'avantage d'une simulation avec des contraintes à priori dans un contexte d'optimisation est que si celles-ci ne sont pas respectées, ce qui peut être déterminé rapidement, la simulation n'est même pas effectuée et l'évaluation n'est pas comptée. Pour cette raison, il est souhaitable de ne pas considérer ces points dans l'analyse. Sachant que pour un certain point et une certaine contrainte, si une fidélité est représentative, toute fidélité plus grande l'est également, il s'ensuit que pour tout $j \in J$, les proportions r_{ij} sont monotone croissantes avec la fidélité.

4.2 Modèle d'optimisation de la matrice de biadjacence

Étant données les estimations précédentes, un modèle d'optimisation est proposé pour déterminer la matrice de biadjacence B qui minimise l'espérance du temps requis par un lancement de l'algorithme 3.2. Dans ce modèle, un dernier paramètre $\varepsilon \in [0, 1]$ que l'utilisateur doit choisir est introduit. Il s'agit d'une borne supérieure posée sur la probabilité estimée

qu'une contrainte c_j avec $j \in J$ soit mal représentée. Le problème (P) est un problème de minimisation du temps espéré. Toutefois, simplement minimiser le temps se résume à assigner toutes les contraintes à la plus petite fidélité. Intuitivement, le modèle devrait minimiser le temps et minimiser la probabilité d'effectuer des interruptions sur des points réalisables. Or, les modèles d'optimisation biobjectifs sont plus complexes. Il est préférable d'uniquement minimiser l'espérance du temps, et d'inclure la contrainte (4.3) qui pose une borne supérieure ε sur la probabilité estimée qu'une c_j soit mal représentée. À chaque lancement de l'algorithme 3.2, des sous-évaluations sont exécutées. Celles-ci correspondent aux variables y_i dans le modèle. Pour chaque fidélité $\phi_i \in \Phi$, $y_i = 1$ si une sous-évaluation est effectuée, et $y_i = 0$ sinon. Durant une sous-évaluation, les c_j qui sont vérifiées sont données par B .

$$\min_{B \in \mathbb{B}^{\ell \times m}, y \in \mathbb{R}^\ell} t_1 y_1 + \sum_{i=2}^{\ell} t_i y_i \prod_{k=1}^{i-1} \prod_{j \in J} (1 - \bar{p}_{kj} B_{kj}) \quad (4.1)$$

$$(P) \quad \text{s.c.} \quad \sum_{i \in I} B_{ij} = 1 \quad \forall j \in J \quad (4.2)$$

$$B_{ij} - \varepsilon \leq r_{ij} \quad \forall i \in I, \forall j \in J \quad (4.3)$$

$$B_{ij} \leq y_i \leq 1 \quad \forall i \in I, \forall j \in J \quad (4.4)$$

$$y_i \leq \sum_{j \in J} B_{ij} \quad \forall i \in I. \quad (4.5)$$

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

La fonction objectif (4.1) est une somme de temps, chacun multiplié par la probabilité que ce temps soit ajouté à l'évaluation. La première sous-évaluation est effectuée avec une probabilité de y_1 , qui vaut 1 ou 0 : on a alors $t_1 y_1$. Le temps t_i avec $i \geq 2$ est ensuite ajouté à l'objectif si $y_i = 1$, donc si une sous-évaluation est effectuée à ϕ_i , et si les sous-évaluations aux fidélités inférieures n'ont pas causé d'interruption. La fonction objectif est alors donnée

par

$$\begin{aligned}
f &= t_1 y_1 \\
&+ t_2 y_2 Pr[\text{pas d'interruption à } \phi_1] \\
&+ t_3 y_3 Pr[\text{pas d'interruption à } \phi_1] Pr[\text{pas d'interruption à } \phi_2] \\
&+ t_4 y_4 Pr[\text{pas d'interruption à } \phi_1] Pr[\text{pas d'interruption à } \phi_2] Pr[\text{pas d'interruption à } \phi_3] \\
&+ \dots \\
&= t_1 y_1 + \sum_{i=2}^{\ell} \left(t_i y_i \prod_{k=1}^{i-1} Pr[\text{pas d'interruption à } \phi_k] \right). \tag{4.6}
\end{aligned}$$

Les indices k dénotent des indices des fidélités inférieures à un certain $i \in I$. La probabilité qu'il n'y ait pas d'interruption à ϕ_k correspond à la probabilité que chaque contrainte assignée à cette fidélité soit satisfaite. Cette probabilité correspond à la multiplication des probabilités que chaque contrainte soit satisfaite individuellement, qui sont estimées par p_{kj} . La formulation

$$P_k(B) := Pr[\text{pas d'interruption à } \phi_k] = \prod_{j \in J} 1 - \bar{p}_{kj} B_{kj} \tag{4.7}$$

est telle que si $B_{kj} = 0$, le j -ième élément de la multiplication est égal à 1, et si $B_{kj} = 1$, signifiant que la contrainte sera vérifiée par l'algorithme, le j -ième élément vaut p_{kj} . En substituant (4.7) dans (4.6), on obtient la forme finale (4.1). À titre illustratif, la solution de l'exemple 3.1 est telle que

$$\begin{aligned}
f &= t_1(1) + t_2(0) \prod_{k=1}^1 P_k(B) + t_3(1) \prod_{k=1}^2 P_k(B) + t_4(1) \prod_{k=1}^3 P_k(B) \\
&= t_1 + 0 + t_3 \left(\prod_{j \in J} 1 - \bar{p}_{1j} B_{1j} \right) \left(\prod_{j \in J} 1 - \bar{p}_{2j} B_{2j} \right) \\
&\quad + t_4 \left(\prod_{j \in J} 1 - \bar{p}_{1j} B_{1j} \right) \left(\prod_{j \in J} 1 - \bar{p}_{2j} B_{2j} \right) \left(\prod_{j \in J} 1 - \bar{p}_{3j} B_{3j} \right) \\
&= t_1 + t_3(p_{11}p_{12})(1) + t_4(p_{11}p_{12})(1)(p_{33}p_{35}) \\
&= t_1 + t_3 p_{11}p_{12} + t_4 p_{11}p_{12}p_{33}p_{35}.
\end{aligned}$$

4.3 Différences entre le modèle et le vrai problème

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

- l'ajout d'une sous-évaluation à $\phi = 1$ si la dernière sous-évaluation n'est pas effectuée à $\phi = 1$ et si le point en évaluation est le meilleur depuis le début de l'optimisation augmente le temps espéré. Or, il est impossible de prévoir à quels moments cette sous-évaluation survient, ce temps est donc impossible à ajouter au modèle ;
- les contraintes à priori ne sont pas considérées dans le modèle, car les points pour lesquels une de ces contraintes n'est pas satisfaite ne sont pas comptés dans l'analyse du comportement des contraintes. Cette pratique est désirable, car ces points ne sont pas comptés non plus lors de l'optimisation. De ce fait, certaines sous-évaluations ont un temps approximativement nul, ce qui diminue l'espérance du temps. En revanche, cette différence n'a aucun impact sur le déroulement de l'optimisation et sur la solution optimale ;
- tel que le modèle est présenté, il est assumé que pour un point quelconque, si la contrainte c_j est satisfaite à une fidélité ϕ_i où $B_{ij} = 1$, elle est satisfaite à toute fidélité supérieure à ϕ_i . Cela n'est pas toujours vrai.

Expliquons ce dernier point en plus amples détails. Le modèle exprime un algorithme qui ne vérifie que les c_j assignées à ϕ_i à chaque sous-évaluation. Toutefois, l'algorithme 3.2 vérifie également les c_j assignées à des fidélités inférieures. Par exemple, dans le problème de l'exemple 3.1, la sous-évaluation à ϕ_4 a lieu si c_3 et c_5 sont satisfaites à ϕ_3 , si c_1 et c_2 sont satisfaites à ϕ_1 (probabilité estimée de $p_{11}p_{12}$) et si elles sont encore satisfaites à ϕ_3 (probabilité estimée de $p_{31}p_{32}$). Or, il est possible que p_{31} et p_{32} ne soient pas indépendants de p_{11} et p_{12} . Pour multiplier ces probabilités, il faut considérer la probabilité p_{31} sachant que $\beta(c_1(x, \phi_1)) = 0$, et la probabilité p_{32} sachant que $\beta(c_2(x, \phi_1)) = 0$ pour un x quelconque. L'inconvénient est que cela complexifie grandement le modèle.

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

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

Dans l'exemple 3.1, la fonction objectif devient

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

La seconde hypothèse stipule que si une contrainte est satisfaite à une fidélité à laquelle

elle est assignée, elle le sera encore pour toute fidélité supérieure. C'est l'hypothèse qui est posée pour déterminer la fonction objectif (4.1) du problème (P) . Il est à noter que plus ε est proche de zéro, plus cette hypothèse est proche de la réalité (à $\varepsilon = 0$ l'hypothèse est valide). Pour un certain point lors d'une évaluation, parmi les fidélités qui ont une haute représentativité, il est rare (au pire, pour une proportion de ε des cas) qu'une contrainte ne donne pas la même valeur de $\beta(\cdot)$ que la vérité. C'est donc l'hypothèse qui est favorable, à moins qu'un grand ε soit choisi.

4.4 Résolution rapide du modèle

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

4.4.1 Réduction 1

La première réduction consiste à omettre les variables de décision y_i , l'équation (4.4) et l'équation (4.5) ainsi qu'à modifier l'équation (4.3). Comme on vise à évaluer la valeur de la fonction objectif de chaque solution restante après les réductions, il n'est pas problématique d'introduire des fonctions non différentiables au problème. De ce fait, les y_i peuvent être calculés à partir d'une matrice B avec

$$y_i(B) = \begin{cases} 0 & \text{si } \sum_{j \in J} B_{ij} = 0 \\ 1 & \text{sinon} \end{cases} \quad \forall i \in I \quad (4.8)$$

plutôt que d'être des variables de décision. Ainsi, les équations (4.4) et (4.5) sont toujours satisfaites, et la matrice B seule constitue une solution au problème. Introduisons la fonction $i(j)$, qui renvoie l'indice de la plus petite fidélité à laquelle la contrainte c_j peut être assignée tout en respectant l'équation (4.3) :

$$i(j) = \min\{i \in I : r_{ij} \geq 1 - \varepsilon\}.$$

L'équation (4.3) est alors équivalente à poser qu'une contrainte c_j ne peut pas être assignée à une fidélité ϕ_i si $i < i(j)$. Posons le problème (P_1) , qui est équivalent au problème (P) :

$$(P_1) \quad \begin{aligned} \min_{B \in \mathbb{B}^{\ell \times m}} \quad & t_1 y_1(B) + \sum_{i=2}^{\ell} \left(t_i y_i(B) \prod_{k=1}^{i-1} P_k(B) \right) \\ \text{s.c.} \quad & \sum_{i \in I} B_{ij} = 1 \quad \forall j \in J \end{aligned} \quad (4.9)$$

$$B_{ij} = 0 \quad \forall i \in I < i(j), \forall j \in J. \quad (4.10)$$

4.4.2 Réduction 2

La seconde réduction vise à réduire le nombre de sommets dans G , et donc la taille de la matrice B . Elle est basée sur le théorème 3, qui indique qu'il existe une solution optimale pour (P_1) où toutes les contraintes sont assignées à des ϕ_i telles que

$$i \in \hat{I} := \bigcup_{j \in J} i(j).$$

Il est alors possible de retirer les ϕ_i où $i \notin \hat{I}$ du modèle d'optimisation sans exclure toutes les solutions optimales. La fonction $i(j)$ dénote un lien direct entre un indice de fidélité i et un indice de contrainte j . En revanche, l'ensemble $\hat{I}$ n'exprime pas ce lien. Le théorème 3 s'intéresse aux solutions où chaque contrainte est assignée à une fidélité dont l'indice correspond à un certain $i(j)$ où $j \in J$, peut importe ce j . Le théorème s'applique sous l'hypothèse que

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

Autrement dit, les p_{ij} sont monotone croissants selon i s'il est possible d'assigner c_j à ϕ_i sans qu'une solution soit non réalisable. Le théorème 3 est démontré, et l'hypothèse est discutée par la suite. Avant cela, l'exemple 4.1 est introduit pour rendre la preuve plus intuitive, avec r la matrice des r_{ij} et p la matrice des p_{ij} . Dans cet exemple, B^a , B^b et B^c sont les trois seules solutions réalisables.

Premièrement, cet exemple démontre qu'assigner chaque contrainte c_j à $i(j)$ ne donne pas nécessairement la solution optimale, car la matrice B^a n'est pas optimale. Deuxièmement, pour B^b , une solution où une contrainte est assignée à un $i \in I \setminus \hat{I}$, la valeur de la fonction objectif est la plus grande. En effet, dans cette situation, la contrainte c_1 pourrait plutôt être assignée à une fidélité plus basse pour diminuer le temps, ou elle pourrait plutôt être assignée à une fidélité plus haute où une autre contrainte est également assignée pour diminuer le nombre de sous-évaluations, et donc diminuer le temps. Le théorème 3 montre qu'il existe toujours

Exemple 4.1 Problème de matrice de biadjacence optimale simple.

$$\begin{array}{l} \varepsilon = 0.05 \\ \ell = 3 \\ m = 2 \end{array} \quad r = \begin{bmatrix} 0.96 & 0 \\ 0.98 & 0 \\ 1 & 1 \end{bmatrix} \quad p = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \quad t = \begin{bmatrix} 6 \\ 8 \\ 10 \end{bmatrix}.$$

De ces données, on détermine

$$\begin{aligned} i(1) &= 1 \\ i(2) &= 3 \\ \hat{I} &= \{1, 3\}. \end{aligned}$$

Pour (P_1) , seulement trois solutions sont réalisables :

$$B^a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, B^b = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, B^c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \quad \text{et } \Omega = \{B^a, B^b, B^c\}$$

$$\begin{aligned} f(B^a) &= t_1 + t_3 p_{11} = 6 + 10/2 = 11 \\ f(B^b) &= t_2 + t_3 p_{21} = 8 + 10/2 = 13 \\ f(B^c) &= t_3 = 10. \end{aligned}$$

une solution optimale où les contraintes sont toutes assignées à des fidélités dont l'indice i appartient à $\hat{I}$. Cette solution est B^c pour l'exemple. En appliquant la réduction (4.4.2), le second élément du vecteur t et la seconde rangée des matrices r , p et B sont omis, et seulement les solutions suivantes, analogues à B^a et B^c , sont considérées :

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{et} \quad \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}.$$

Lemme 1

Soit B une solution réalisable pour (P_1) . S'il existe un indice de fidélité $i' \in I \setminus \hat{I}$ et un indice de contrainte $j \in J$ tel que $B_{i'j} = 1$, alors la matrice B' où

$$B'_{ij} = \begin{cases} 0 & \text{si } i = i' \\ 1 & \text{si } i = i' - 1 \text{ et } B_{i'j} = 1 \\ B_{ij} & \text{sinon} \end{cases} \quad \forall i \in I, \forall j \in J \quad (4.12)$$

est réalisable pour (P_1) .

Démonstration

L'équation (4.9) est satisfaite par B' , car elle est satisfaite par B et $B'_{i'-1j} + B'_{i'j} = 1 = B_{i'-1j} + B_{i'j}$ pour tout $j \in J$. Concernant l'équation (4.10),

$$\begin{aligned} B_{ij} &= 0 & \forall i \in I < i(j), \forall j \in J & \quad \text{car } B \text{ est réalisable} \\ \implies i' > i(j) & & \forall j \in J \text{ où } B_{i'j} = 1 \\ \iff i' - 1 \geq i(j) & & \forall j \in J \text{ où } B_{i'j} = 1 \\ \implies B'_{ij} = B_{ij} = 0 & & \forall i \in I < i(j), \forall j \in J \quad \text{par (4.12)} \\ \implies B' \text{ est réalisable pour } (P_1). \end{aligned} \quad (4.13)$$

□

Lemme 2

Soit B une solution réalisable pour (P_1) . S'il existe un indice de fidélité $i' \in I \setminus \hat{I}$ et un indice de contrainte $j \in J$ tel que $B_{i'j} = 1$, alors la matrice B' donnée par (4.12) satisfait $f(B') \leq f(B)$.

Démonstration

La sommation de la fonction objectif (4.1) peut être divisée en quatre termes : $T_{<i'-1}(B)$, $T_{i'-1}(B)$, $T_{i'}(B)$ et $T_{>i'}(B)$. Ceux-ci correspondent respectivement aux indices de fidélité inférieurs à $i' - 1$, à $i' - 1$, à i' et aux indices de fidélité supérieurs à i' :

$$\begin{aligned} f(B) &= \left(t_1 y_1 + \sum_{i=2}^{i'-2} t_i y_i \prod_{k=1}^{i-1} P_k \right) + t_{i'-1} y_{i'-1} \prod_{k=1}^{i'-2} P_k + t_{i'} y_{i'} \prod_{k=1}^{i'-1} P_k + \sum_{i=i'+1}^{\ell} t_i y_i \prod_{k=1}^{i-1} P_k \\ &= T_{<i'-1}(B) + T_{i'-1}(B) + T_{i'}(B) + T_{>i'}(B). \end{aligned}$$

Dans cette équation, les $y_i(B)$ et les $P_k(B)$ sont notés y_i et P_k pour alléger la notation.

Pour le premier terme,

$$T_{<i'-1}(B) = T_{<i'-1}(B') \quad \text{par (4.12).}$$

Pour le dernier terme, sachant que la seule différence entre B et B' est que les $p_{i'j}$ pour B sont devenus des $p_{i'-1j}$ dans B' ,

$$T_{>i'}(B) \geq T_{>i'}(B') \quad \text{car (4.11).} \quad (4.14)$$

En ce qui concerne les deux termes du centre, le développement dépend de $y_{i'-1}(B)$. Si $y_{i'-1}(B) = 0$:

$$\begin{aligned} T_{i'-1}(B) + T_{i'}(B) &= T_{i'}(B) && \text{car } y_{i'-1}(B) = 0 \\ &= t_{i'} \prod_{k=1}^{i'-1} P_k(B) \\ &= t_{i'} \prod_{k=1}^{i'-2} P_k(B') \\ &\geq t_{i'-1} \prod_{k=1}^{i'-2} P_k(B') && \text{car } t_i \text{ est monotone croissant selon } i \\ &= T_{i'-1}(B') \\ &= T_{i'-1}(B') + T_{i'}(B') && \text{car } y_{i'}(B') = 0. \end{aligned}$$

Si plutôt, $y_{i'-1}(B) = 1$, $T_{i'-1}(B) = T_{i'-1}(B')$ car la sous-évaluation à $\phi_{i'-1}$ et celles aux fidélités inférieures sont inchangées, et $T_{i'}(B) > T_{i'}(B') = 0$ car $y_{i'}(B') = 0$. Il en résulte que peu importe la valeur de $y_{i'-1}(B)$, les termes de $f(B)$ sont supérieurs ou égaux à ceux de $f(B')$. Conséquemment,

$$f(B) \geq f(B').$$

□

Théorème 3

Il existe une solution optimale pour (P_1) où toutes les contraintes sont assignées à des fidélités ϕ_i où $i \in \hat{I}$.

Démonstration

L'argmin de (P_1) peut ne contenir que des solutions où toutes les contraintes sont assignées à des fidélités ϕ_i où $i \in \hat{I}$. Le théorème est alors trivial. Autrement, il existe une solution B^*

dans l'argmin de (P_1) qui est telle que l'ensemble

$$I' := \{i \in I \setminus \hat{I} : y_i(B^*) = 1\}$$

n'est pas vide. Pour un certain $i' \in I'$, il est possible de déterminer une nouvelle solution où toutes les contraintes assignées à $\phi'_{i'}$ dans B^* sont plutôt assignées à $\phi'_{i'-1}$, et où les autres assignations sont inchangées. Le lemme 1 indique que cette nouvelle solution est réalisable, et le lemme 2 indique que cette solution donne un objectif de valeur égale ou meilleure à B^* . Cela implique qu'elle appartient également à l'argmin. À partir de cette nouvelle solution, l'ensemble I' peut être recalculé. Tant qu'il n'est pas vide, l'exercice peut être répété. Par (4.13), le nombre maximum d'itérations est de $\max\{\ell - i(j) : j \in J\}$, après quoi I' est vide. Dès lors,

$$\exists B^* \in \operatorname{argmin}_{B \in \Omega} \{f(B)\} \text{ où } B_{ij}^* = 0 \forall i \in I \setminus \hat{I}, \forall j \in J.$$

□

L'hypothèse (4.11) assure que l'équation (4.14) du lemme 2 est vraie. Considérons l'exemple 4.2 où l'hypothèse est fausse, avec r la matrice des r_{ij} et p la matrice des p_{ij} . Dans cet exemple, B est une solution telle que c_1 est assignée à $\phi_{i'}$ où $i' = 2$ et B' correspond à (4.12).

Exemple 4.2 Contrexemple à l'hypothèse (4.11).

$$\begin{array}{l} \varepsilon = 0.05 \\ \ell = 3 \\ m = 2 \end{array} \quad r = \begin{bmatrix} 0.95 & 0.7 \\ 1 & 0.7 \\ 1 & 1 \end{bmatrix} \quad p = \begin{bmatrix} 0.53 & 0.4 \\ 0.5 & 0.4 \\ 0.5 & 0.4 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B' = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

L'hypothèse est fausse puisque $p_{11} > p_{21}$ et B et B' sont réalisables. En reprenant la notation du lemme 2, la valeur de la fonction objectif pour les solutions sont les suivantes.

$$\begin{aligned} f(B) &= \underbrace{0}_{T_{<i'-1}(B)} + \underbrace{0}_{T_{i'-1}(B)} + \underbrace{t_2}_{T_{i'}(B')} + \underbrace{t_3 p_{21}}_{T_{>i'}(B)} \\ f(B') &= \underbrace{0}_{T_{<i'-1}(B')} + \underbrace{t_1}_{T_{i'-1}(B')} + \underbrace{0}_{T_{i'}(B')} + \underbrace{t_3 p_{11}}_{T_{>i'}(B')}. \end{aligned}$$

On a alors que

$$T_{>i'}(B) = t_3 p_{21} < t_3 p_{11} = T_{>i'}(B'),$$

ce qui contredit (4.14).

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

$$\begin{aligned} f(B^b) \geq f(B^a) &\iff t_2 + t_3 p_{21} \geq t_1 + t_3 p_{11} \\ &\iff t_2 - t_1 \geq (p_{11} - p_{21}) t_3 \end{aligned} \tag{4.15}$$

Si l'hypothèse était satisfaite ($p_{11} \leq p_{21}$), l'inégalité (4.15) serait trivialement satisfaite car les t_i sont positifs et monotone croissants selon i . Dans l'exemple donné, l'inégalité est vraie si le temps réduit par l'assignation de c_1 à une fidélité plus basse ($t_2 - t_1$) est plus significatif que la perte de probabilité d'effectuer une interruption qui coupe t_3 secondes $((p_{11} - p_{21}) t_3)$.

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

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

Pour revenir à l'exemple, si p_{11} n'était pas donné, il est tout de même connu que $p_{11} \in$

[0.45, 0.55]. La perte de probabilité $p_{11} - p_{21}$ d'effectuer une interruption à ϕ_3 est alors d'au plus $\varepsilon = 0.05$. Pour résumer, l'hypothèse (4.11) assure que la réduction (4.4.2) n'exclue pas de solution optimale. Aussi, même si l'hypothèse est fausse, la réduction n'exclue pas de solution optimale si les nouvelles solutions déterminées au théorème 3 sont telles que le gain en temps est plus significatif que la perte de probabilité d'effectuer des interruptions, qui est bornée supérieurement par ε .

4.4.3 Réduction 3

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

$$\hat{J} := \{j \in J : c_j \text{ n'est pas une contrainte à priori}\}. \quad (4.16)$$

4.4.4 Réduction 4 et résolution

La quatrième réduction consiste à ne pas considérer les solutions non réalisables. Avec les réductions (4.4.2) et (4.4.3), on pose le problème (P_1) où les rangées n'appartenant pas à $\hat{I}$ des matrices B , p et r et les éléments d'indices n'appartenant pas à $\hat{I}$ du vecteur t sont omis, et où les colonnes n'appartenant pas à $\hat{J}$ des matrices B , p et r sont omises. Nommons ce problème le problème (P_1) réduit. L'équation (4.3) indique que les arêtes $[c_j, \phi_i]$ et $[\phi_i, c_j]$ ne peuvent pas exister si $r_{ij} < 1 - \varepsilon$. L'équation (4.2) indique que les sommets correspondant aux contraintes ont un degré égal à 1. Il est possible de lister exhaustivement les solutions qui respectent ces deux équations. Pour chaque solution dans cette liste, l'objectif (4.1) est calculé, et les solutions qui donnent la plus petite valeur de l'objectif sont optimales. À partir d'une de ces solutions, la matrice B^* de dimension $\ell \times m$ est reconstruite en y ajoutant les colonnes $J \setminus \hat{J}$ et les rangées $I \setminus \hat{I}$, et en donnant à chacun de ces nouveaux éléments une valeur nulle. Cette matrice de biadjacence permet de déterminer E , et la dernière étape de l'algorithme 3.1 peut être lancée.

4.5 Algorithme d'optimisation avec contraintes hiérarchisées détaillé

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

L'algorithme 4.1 peut être adapté au cas où l'utilisateur ne connaît pas de bon point de départ. D'abord, les bornes de l'hypercube latin sont définies par X , peu importe la valeur de ρ . Après avoir effectué les évaluations, il suffit de définir le meilleur point de l'échantillon comme étant x^0 , puis de considérer uniquement les points de l'hypercube latin qui sont à l'intérieur des bornes inférieures l^{cen} et des bornes supérieures u^{cen} pour le calcul des r_{ij} , des p_{ij} et des t_i . Toutefois, cette méthode ne fonctionne que pour les problèmes de petite dimension ou avec un grand ρ . Autrement, il est possible que le nombre de points contenus par l^{cen} et u^{cen} soit trop petit. Si, par exemple, un problème est de dimension $n = 3$ et que l'utilisateur choisit $\rho = 1/4$, le volume de l'échantillon considéré pour l'analyse du comportement des contraintes est le volume de l'hypercube divisé par deux ($2\rho = 1/2$) dans trois axes. De façon générale, le volume de l'hypercube est divisé par au moins $(2\rho)^{-n}$. Le volume est réduit davantage si $x_i^0 + \rho(u_i - l_i) > u_i$ ou si $x_i^0 - \rho(u_i - l_i) < l_i$ pour un $i \in I$. De ce fait, il est recommandé de poser $\rho = 1$ pour les problèmes de haute dimension où un point de départ n'est pas connu.

Algorithme 4.1 : Optimisation avec contraintes hiérarchisées (détaillé)

Fonction $Optimisation(x^0, BB, \rho, \Phi, \varepsilon, n_H)$

Définition des paramètres d'entrée

- x^0 : point de départ de l'optimisation (optionnel).
- BB : objet contenant l'espace des variables X , les n variables, les m contraintes, la fonction objectif f et la fonction *Sous-évaluation*.
- ρ : facteur de rapprochement des bornes de l'hypercube latin.
- Φ : ensemble de fidélités ordonné et se terminant par 1.
- ε : borne supérieure sur la probabilité estimée qu'une contrainte soit mal représentée.
- n_H : taille de l'hypercube latin.

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

- Déterminer les bornes l^{cen} et u^{cen} à partir de X , ρ et x^0 .
- Déterminer les points de H , l'hypercube latin de n_H points borné par l^{cen} et u^{cen} .
- Évaluer les points de H à chaque fidélité de Φ , en parallélisant les évaluations autant que possible.
- Calculer les estimations de probabilité r_{ij} et p_{ij} , ainsi que les temps moyens t_i .

2. Calcul d'une matrice de biadjacence optimale.

- Déterminer $\hat{I}$ avec ε et $\hat{J}$.
- Résoudre le problème (P_1) réduit avec une recherche exhaustive sur Ω .
- Construire $B^* \in \mathbb{B}^{\ell \times m}$ à partir d'une solution optimale, puis déterminer E .
- Construire le graphe biparti $G = (V_1, V_2, E)$ où $V_1 = \Phi$, $V_2 = \{c_1, c_2, \dots, c_m\}$.

3. Optimisation de la boîte noire.

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

Renvoyer les sorties du solveur.

CHAPITRE 5 BANC D’ESSAI ET RÉSULTATS INTERMÉDIAIRES

Le cinquième chapitre présente les résultats numériques des deux premières étapes de l’algorithme 4.1. Comme le projet PRIAD n’a pas atteint la phase de l’optimisation de boîte noire, les tests sont effectués avec la famille de boîtes noires **solar** version 0.5.3 que l’on peut trouver à <https://github.com/bbopt/solar>. Ces boîtes noires sont décrites par Lemyre Garneau (2015), et elles ont été développées pour offrir un banc d’essai pour les algorithmes d’optimisation de boîtes noires. La famille de boîtes noires **solar** simule une centrale de production électrique solaire thermodynamique. Les problèmes **solar** faisant l’objet des tests sont d’abord présentés, puis les analyses du comportement des contraintes ainsi que les calculs des matrices de biadjacence sont discutés.

5.1 Banc d’essai et implémentations

Quatre problèmes contraints de la famille **solar** comportent un aspect multifidélité : **solar2**, **solar3**, **solar4** et **solar7**. Ces différentes instances constituent des simulations de différentes parties d’une centrale solaire thermodynamique. Leurs différentes caractéristiques sont présentées au tableau 5.1. La colonne m_{ap} dénote le nombre de contraintes à priori, $m_{\text{multi-}\phi}$ dénote le nombre de contraintes affectées par la multifidélité, f_{an} indique si l’objectif est formulé analytiquement et $f_{\text{multi-}\phi}$ indique si l’objectif est affecté par la multifidélité. Une contrainte affectée par la multifidélité est nécessairement simulée, et est donc de simulation.

TABLEAU 5.1 Caractéristiques de quatre boîtes noires **solar** multifidélités.

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

Une remarque importante est que parmi les fonctions objectif des quatre boîtes noires, seulement celle de **solar7** est affectée par la multifidélité. Conséquemment, il est imposé pour **solar7** que la dernière sous-évaluation effectuée par l’algorithme 3.2 soit avec une fidélité

de 1. Autrement, il est possible que l'information donnée au solveur sur f pour les points réalisables soit erronée. En imposant une sous-évaluation à une fidélité de 1, il est garanti que la valeur de $f(x)$ donnée au solveur pour tout x réalisable soit la vérité. Au niveau de l'implémentation, l'équation (4.8) doit être changée pour

$$y_i(B) = \begin{cases} 0 & \text{si } \sum_{j \in J} B_{ij} = 0 \text{ et } i \neq \ell \\ 1 & \text{sinon} \end{cases} \quad \forall i \in I$$

Dans ce cas, la solution optimale de (P_1) est la matrice de biadjacence qui minimise l'espérance du temps sachant que la sous-évaluation à $\phi = 1$ a toujours lieu.

L'instance **solar7** contient peu de contraintes, et celles-ci sont faciles à satisfaire. Ce problème permet d'observer comment la méthode d'optimisation se comporte lorsqu'elle a peu de moyens permettant d'effectuer des interruptions, car elle s'appuie sur les contraintes. De plus, n'ayant que 7 variables, **solar7** est un problème que les solveurs existants peuvent résoudre efficacement. Les instances **solar2** et **solar3** comportent des contraintes difficiles à satisfaire. Il s'agit d'instances où la méthode testée a du potentiel. En ce qui concerne **solar4**, les contraintes sont significativement plus difficiles à satisfaire. Cette boîte noire permet de tester la méthode dans un environnement où simplement trouver un point réalisable est d'une grande difficulté. Aussi, comportant 29 variables, **solar4** est un problème qui est significativement plus difficile pour les solveurs existants. Ces propos sont supportés par le tableau 5.2, qui montre la quantité de points qui satisfont les contraintes à priori, et la quantité de points qui satisfont toutes les contraintes de quatre hypercubes latins de 10^4 points. Ces points sont évalués à $\phi = 1$ seulement. Le tableau montre également le temps moyen d'évaluation des points qui respectent les contraintes à priori correspondant à t_ℓ pour le même hypercube latin, pour chaque instance.

TABLEAU 5.2 Réalisabilité et temps de 10^4 points d'hypercubes latins pour 4 instances **solar**.

Instance	Réalisables à priori	Réalisables	Temps moyen [s]
solar2	1853	1	29.1
solar3	894	1	4.61
solar4	10	0	7.36
solar7	4608	1354	10.26

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

Pour l’analyse du comportement des contraintes ainsi que le calcul de la matrice de biadjacence B , différentes configurations sont créées en variant certains paramètres. La section 4.1 mentionne que l’utilisateur doit choisir un ensemble Φ incluant 1, une valeur de ρ , et une taille d’hypercube latin n_H . Lorsque les évaluations aux basses fidélités sont significativement plus rapides que les autres (ce qui est le cas pour les boîtes noires **solar**), il est peu coûteux d’ajouter de nombreuses basses fidélités. L’ensemble

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

est choisi. À des fins de comparaisons, trois différents hypercubes latins sont déterminés, et ce, pour chacun des quatre problèmes **solar**. Un total de douze configurations sont donc testées, et les caractéristiques de chacune sont illustrées au tableau 5.3. Lorsque $\rho < 1$, un point réalisable connu est choisi comme centre de l’hypercube latin tel que décrit à la section 4.1. Un tel échantillon est nommé un hypercube latin centré. Les bornes définissant chaque hypercube sont présentées aux tableaux B.1 et B.2.

TABLEAU 5.3 Configurations testées pour l’analyse du comportement des contraintes.

Paramètre	Configuration											
n_H	10^4	10^3	10^3	10^4	10^3	10^3	10^4	10^3	10^3	10^4	10^3	10^3
ρ	1	1	1/4	1	1	1/10	1	1	1/20	1	1	1/4
Instance	solar2			solar3			solar4			solar7		

5.2 Analyse du comportement des contraintes

Suite aux évaluations de chaque configuration, la proportion de satisfaction p_{ij} et la proportion de représentativité r_{ij} sont calculées pour chaque $(i, j) \in (I, J)$, et le temps d’évaluation moyen t_i est calculé pour chaque i dans I .

5.2.1 solar2

Débutons avec les trois configurations de **solar2**. Ces calculs correspondent à la première étape de l'algorithme 4.1. Les proportions r_{ij} sont illustrées sous forme de tableaux à la figure 5.1, où seulement les contraintes affectées par la fidélité sont présentes. Pour toute autre contrainte, le calcul de r_{ij} donne 1 pour toutes les fidélités.

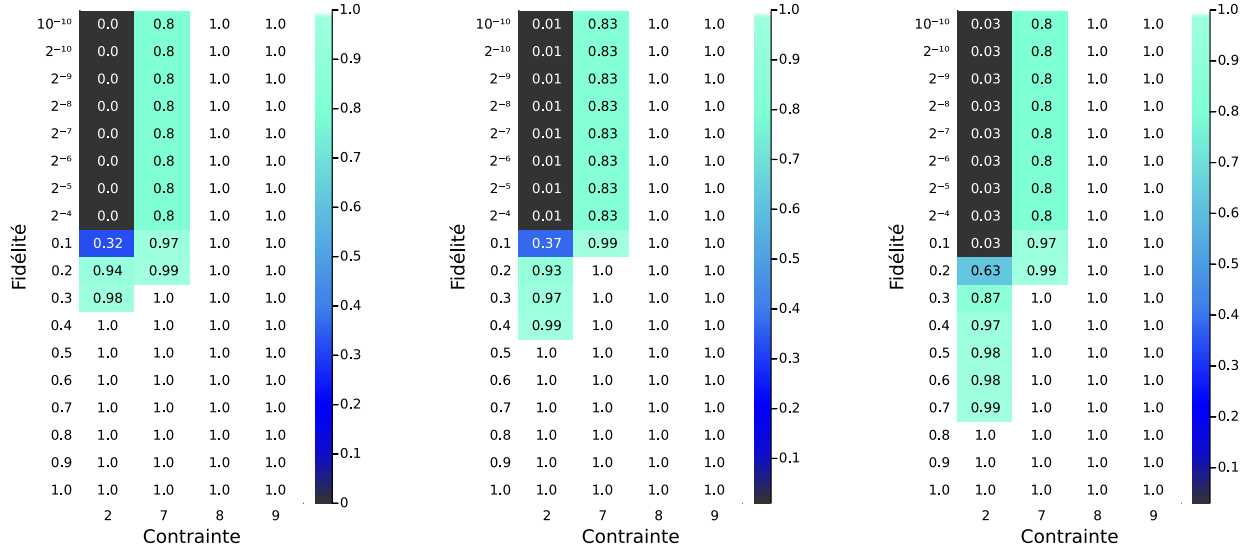


FIGURE 5.1 Valeurs des r_{ij} de **solar2** avec différentes configurations d'hypercubes latins.

Les quatre contraintes affectées par la fidélité sont c_2, c_7, c_8 et c_9 . Une première remarque est que les fidélités comprises entre 2^{-10} et 2^{-4} inclusivement n'apportent aucune information intéressante, elles pourraient être ignorées sans impacter la matrice de biadjacence optimale. Une seconde remarque est que toutes les fidélités de Φ sont représentatives pour les contraintes c_8 et c_9 pour tous les points échantillonnés. Cela signifie que la valeur de ces contraintes peut varier avec la fidélité pour le même point, mais jamais de sorte à obtenir une différente valeur de β . Par exemple, dans l'hypercube latin de 10^4 points, le point x^0 est tel que $c_8(x^0, 10^{-10}) = -59.962561$, $c_8(x^0, 0.1) = -59.827975$ et $c_8(x^0, 1) = -59.324137$. En observant aussi la valeur de $c_8(x^0, \phi)$ aux autres fidélités, on constate que $\beta(c_8(x^0, \phi)) = 0$ pour tout $\phi \in \Phi$. Le même phénomène a lieu pour c_9 et pour les 10^4 points de l'échantillon, d'où la colonne de 1.0 pour c_8 et c_9 . Il est à noter qu'ici, la valeur de β est de 0 pour tout point et pour les deux contraintes, mais la fonction binaire β peut varier d'un point à l'autre. L'important est qu'elle soit identique à toute fidélité pour le même point. Ces contraintes sont un excellent levier pour la méthode d'optimisation avec contraintes hiérarchisées, car elles peuvent être vérifiées

avec une sous-évaluation presque instantanée. Seulement les contraintes c_2 et c_7 sont telles que certaines fidélités ne sont parfois pas représentatives. Les trois configurations donnent des résultats plutôt similaires. Avec $\varepsilon = 0.05$, $\phi_{i(2)} = 0.3$ en (a), $\phi_{i(2)} = 0.3$ en (b) et $\phi_{i(2)} = 0.4$ en (c). Aussi, $\phi_{i(7)} = 0.1$ pour les trois configurations. Le choix de l'hypercube latin a donc peu d'impact sur la représentativité observée lors de l'analyse.

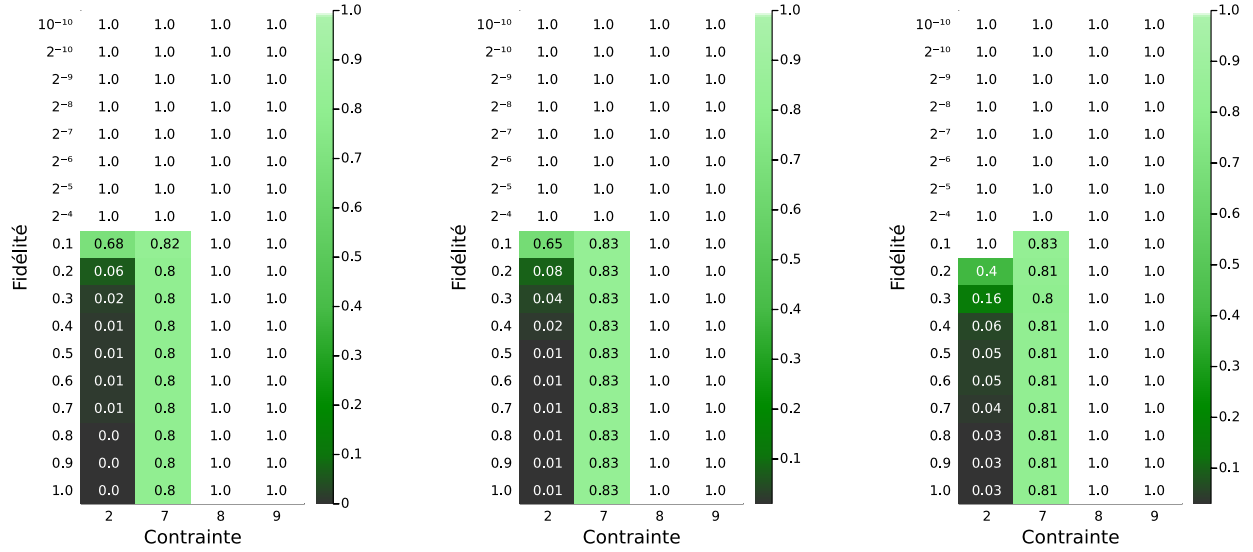


FIGURE 5.2 Valeurs des p_{ij} de *solar2* avec différentes configurations d'hypercubes latins.

Un constat similaire est effectué avec les proportions de satisfaction p_{ij} illustrées à la figure 5.2. En effet, les proportions diffèrent peu d'une configuration à l'autre. Aussi, pour les trois configurations, l'hypothèse (4.11) n'est pas respectée. Cela est toutefois peu problématique. Expliquons pourquoi en observant la contrainte c_2 l'hypercube latin centré (c), en rappelant d'abord que l'hypothèse de croissance monotone des p_{ij} ne concerne que les paires (i, j) où aucune contrainte de (P_1) n'est violée lorsque $B_{ij} = 1$. Cela implique que pour chaque contrainte c_j , les indices de fidélité $i < i(j)$ ne sont pas concernés. Comme $\phi_{i(2)} = 0.4$, seulement les p_{i2} où $\phi \geq 0.4$ sont concernés par l'hypothèse. Le plus grand écart entre deux de ces p_{i2} est 0.03. Pour la contrainte c_7 , avec $\phi_{i(7)} = 0.1$, cet écart maximal est de 0.03 également. Il est expliqué à la fin de la section 4.4.2 que plus cet écart est petit, moins il est probable que la seconde réduction exclue une solution optimale pour (P_1) . En rappelant que 0.03 est une proportion, donc un nombre entre 0 et 1, il est juste de conclure que le fait que l'hypothèse (4.11) n'est pas respectée est peu significatif pour *solar2*. Aussi, l'écart maximal parmi les contraintes en (a) est de 0.02 et celui en (b) est de 0.03. Pour les proportions de satisfaction p_{ij} , aucune information utile n'est apportée par les fidélités 2^{-4} à

2^{-10} , elles pourraient encore une fois être ignorées sans problèmes.

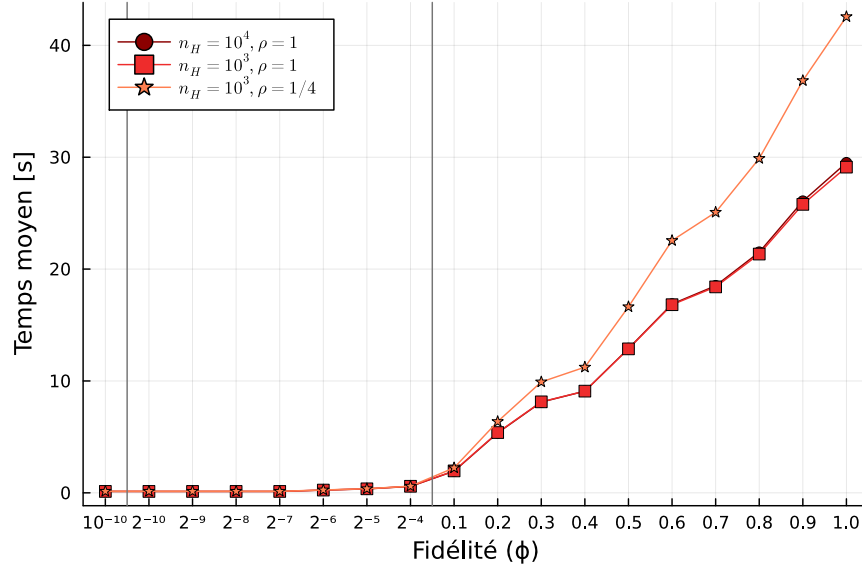


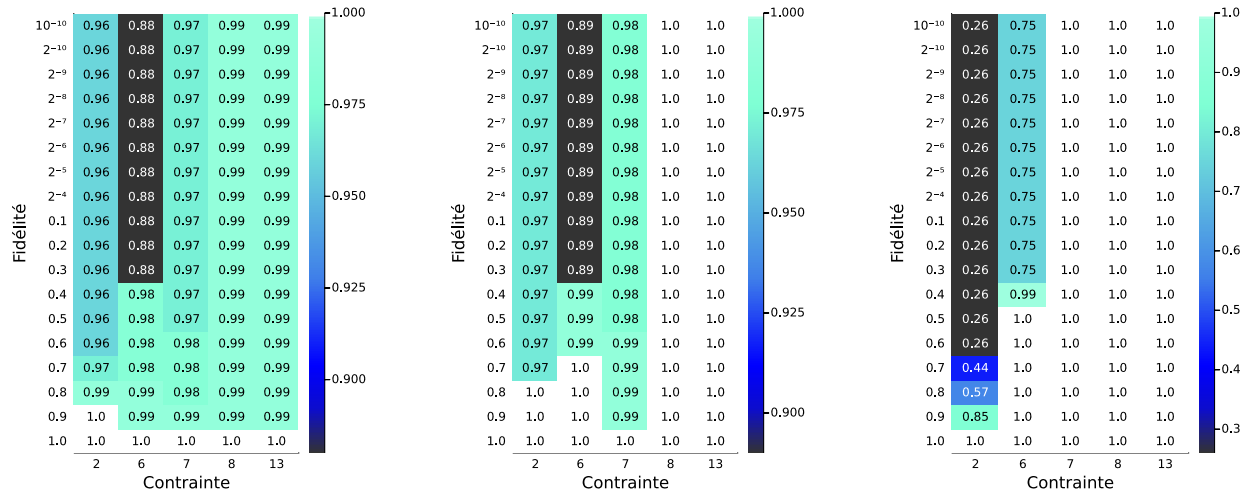
FIGURE 5.3 Valeurs des t_i de *solar2* avec différentes configurations d’hypercubes latins.

Ce dernier constat est aussi vrai pour les temps moyens d’exécution de la boîte noire à chaque fidélité illustrés à la figure 5.3. Dans cette dernière, des lignes verticales indiquent les changements d’échelle sur l’axe des fidélités. Les deux configurations avec $\rho = 1$ donnent des temps presque identiques, et la configuration avec $\rho = 1/4$ donne des temps plus élevés. Cela suggère que le point de départ choisi autour duquel les points sont échantillonnés pour l’hypercube latin centré se situe dans un sous-ensemble de l’espace des paramètres où les évaluations sont plus longues. Cette observation n’est toutefois pas pertinente pour le modèle de matrice de biadjacence optimale. Ce qui importe pour ce modèle, ce sont les écarts relatifs entre les différentes fidélités, et ces écarts sont presque identiques pour toutes les configurations. Par exemple, entre les fidélités 0.3 et 0.4, l’écart de temps est moindre par rapport à l’écart de temps entre 0.4 et 0.5. On conclut alors que le comportement des contraintes est plutôt indépendant de l’hypercube latin effectué pour *solar2*. Un échantillon de 10^3 points est suffisant pour l’analyse, et l’impact du choix de l’hypercube latin sur le déroulement de l’algorithme 3.2 est minime.

5.2.2 solar3

L’instance *solar3* comprend cinq contraintes affectées par la fidélité : c_2 , c_6 , c_7 , c_8 et c_{13} . Les proportions r_{ij} sont illustrées à la figure 5.4. Premièrement, tout comme pour l’instance *solar2*, la valeur de certaines contraintes varie avec la fidélité sans que la valeur de β ne change en (b) et en (c). Deuxièmement, un changement significatif survient entre les hypercubes

latins qui échantillonnent l'espace des paramètres et celui qui échantillonne seulement autour d'un point de départ. En (a) et en (b), la contrainte c_2 est représentative pour 96% et 97% des points échantillonnés respectivement pour les fidélités inférieures ou égales à 0.6. Cette proportion est plutôt de 26% pour l'hypercube latin centré. La contrainte c_6 est également un peu plus rarement représentative aux basses fidélités avec cet hypercube. Un utilisateur qui échantillonne avec $\rho = 1$ sera porté à croire qu'il peut vérifier la valeur de c_2 aux plus basses fidélités sans problèmes, ce qui est faux selon l'analyse en (c). D'un autre côté, les quatre autres contraintes exhibent des comportements plutôt similaires. Dernièrement, les résultats en (a) et en (b) sont presque identiques. Avec $\rho = 1$, effectuer 10^4 évaluations n'apporte peu d'information supplémentaire par rapport à 10^3 évaluations.



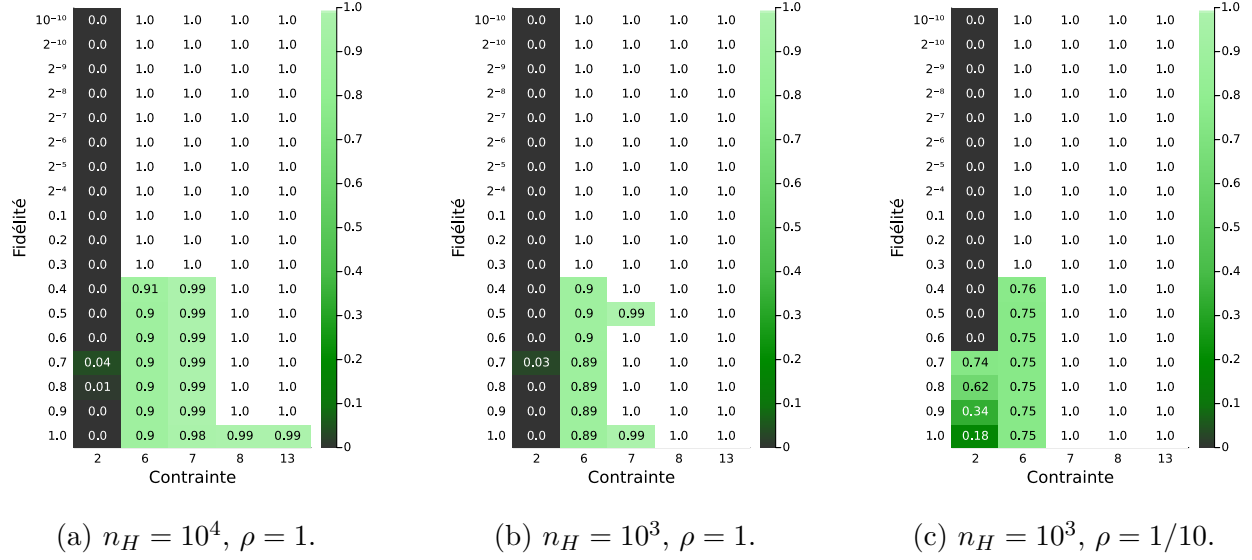
(a) $n_H = 10^4, \rho = 1$.

(b) $n_H = 10^3, \rho = 1$.

(c) $n_H = 10^3, \rho = 1/10$.

FIGURE 5.4 Valeurs des r_{ij} de solar3 avec différentes configurations d'hypercubes latins.

Concernant les proportions de satisfaction illustrées à la figure 5.5, un phénomène intéressant est observé avec la contrainte c_2 . En effet, il semble que pour les trois échantillons, la contrainte n'est pas satisfaite pour tout point et pour toute fidélité inférieure ou égale à 0.6. À $\phi = 0.7$, le comportement de la simulation change, et la contrainte est satisfaite pour quelques rares points, sauf pour l'hypercube centré, où c'est le cas pour 74% des points. La proportion de satisfaction est ensuite décroissante pour les fidélités supérieures ou égales à 0.7. Cette situation illustre bien pourquoi la représentativité est définie de sorte que pour un point donné, même si une fidélité ϕ_i identifie correctement si la contrainte est satisfaite à $\phi = 1$, ϕ_i n'est pas représentative s'il existe une fidélité supérieure différente de 1 qui n'identifie pas correctement si la contrainte est satisfaite. Prenons en exemple les points x^3 , x^6 et x^{15} de l'échantillon où $\rho = 1/10$. Les valeurs de $c_2(x, \phi)$ pour ces trois points pour toute

FIGURE 5.5 Valeurs des p_{ij} de solar3 avec différentes configurations d'hypercubes latins.

$\phi \in \Phi$ sont illustrées au tableau 5.4.

TABLEAU 5.4 Valeur de c_2 à trois points appartenant à l'hypercube latin centré de solar4.

Fidélité $\phi \in \Phi$	Valeur de $c_2(x^3, \phi)$	Valeur de $c_2(x^6, \phi)$	Valeur de $c_2(x^{15}, \phi)$
< 0.6	100	100	100
0.7	0	0	33.5
0.8	0	0	63.2
0.9	0	2.13	75.6
1	1.996	16.31	79.2

Dans chaque cas, la valeur de la contrainte c_2 est 100 pour toute fidélité inférieure ou égale à 0.6. Une analyse approfondie révèle que c'est le cas pour tout point qui respecte les contraintes a priori, c'est-à-dire, pour tout point où la simulation a lieu. Cela semble signifier que le calcul de c_2 requiert un élément de la simulation coûteux en temps, et que cet élément n'est pas calculé si $\phi \leq 0.6$. Le cas échéant, une valeur par défaut de 100 est renvoyée (il est à noter qu'il s'agit d'une supposition). Cela implique que si $c_2(x^3, 10^{-10})$, $c_2(x^6, 10^{-10})$ et $c_2(x^{15}, 10^{-10})$ identifient correctement que c_2 n'est pas satisfaite, ce n'est que

par coïncidence. Le principe de représentativité est qu'une fidélité ϕ_i est dite représentative pour c_j et x s'il est estimé que lorsqu'un phénomène fait en sorte que c_j n'est pas satisfaite à ϕ_i , ce même phénomène fait également en sorte que c_j n'est pas satisfaite à $\phi = 1$, et vice versa si la contrainte est satisfaite en vérité. De toute évidence, le phénomène qui explique que $\beta(c_2(x^6, 10^{-10})) = 1$ n'est pas le même que celui expliquant que $\beta(c_2(x^6, 1)) = 1$. Les deux résultats sont indépendants, d'où la coïncidence. Les fidélités représentatives pour ce point sont plutôt 0.9 et 1. Ce principe de représentativité possède une valeur pratique. Si l'on assignait c_2 à $\phi = 10^{-10}$ par exemple, l'algorithme 3.2 renverrait systématique un point non réalisable au solveur après la première sous-évaluation pour tous les points de l'optimisation, et celle-ci se terminerait sans avoir obtenu de solution réalisable. Il est incorrecte d'énoncer que $\phi = 10^{-10}$ est une fidélité représentative pour c_2 à tous les points x où $\beta(c_2(x, 1)) = 1$.

Pourtant, les analyses des deux autres configurations suggèrent le contraire. Cela s'explique par le fait que pour tous les points évalués lors de ces analyses, un des quatre comportements suivants est observé :

- la valeur de c_2 est nulle pour toute fidélité supérieure ou égale à 0.7. Tous les points où c_2 est respectée sont dans cette catégorie ;
- la valeur de c_2 est nulle pour les fidélités 0.7, 0.8 et 0.9. À $\phi = 1$, elle prend une valeur supérieure à 0, mais qui est petite. Le point x^3 en est un exemple ;
- à $\phi = 0.7$, la valeur de c_2 tombe à 0, puis tend vers une valeur non nulle en augmentant la fidélité. Le point x^6 en est un exemple ;
- la valeur de c_2 est supérieure à 0 pour toute fidélité. Cela survient le plus souvent pour les x où $c_2(x, 1)$ vaut 100 ou un peu moins. Le point x^{15} en est un exemple.

Cela implique que pour un point x , il est identifié que les fidélités inférieures à 0.7 ne sont pas représentatives pour c_2 si $c_2(x, 0.7) = 0$. Cela ne survient pas pour les points de la dernière catégorie, qui contient les points qui ne respectent pas c_2 par la plus grande marge. La figure 5.5 révèle que à la fidélité $\phi = 0.7$, c_2 vaut 0 pour 74% des points en (c), pour 3% des points en (b) et pour 4% des points en (a). On en conclut que les échantillons où $\rho = 1$ contiennent presque uniquement des points qui ne respectent pas c_2 par une grande marge. Inversement, l'hypercube latin centré comprend une majorité de points qui ne respectent pas c_2 par une plus petite marge (ainsi que plus de points où la contrainte c_2 est satisfaite). Ces points sont alors d'une grande importance.

Finalement, avec $\varepsilon = 0.05$, l'hypothèse (4.11) n'est pas respectée en (a), en (b) et en (c), mais les écarts maximaux entre les p_{ij} concernés sont seulement de 0.04, 0.03 et 0.01 respectivement. Les temps moyens pour **solar3** sont donnés par la figure 5.6. Les courbes illustrent des relations entre les temps similaires. Le choix de la configuration a peu d'impact en ce qui a trait au temps. Pour conclure, en considérant les proportions de représentativité et de

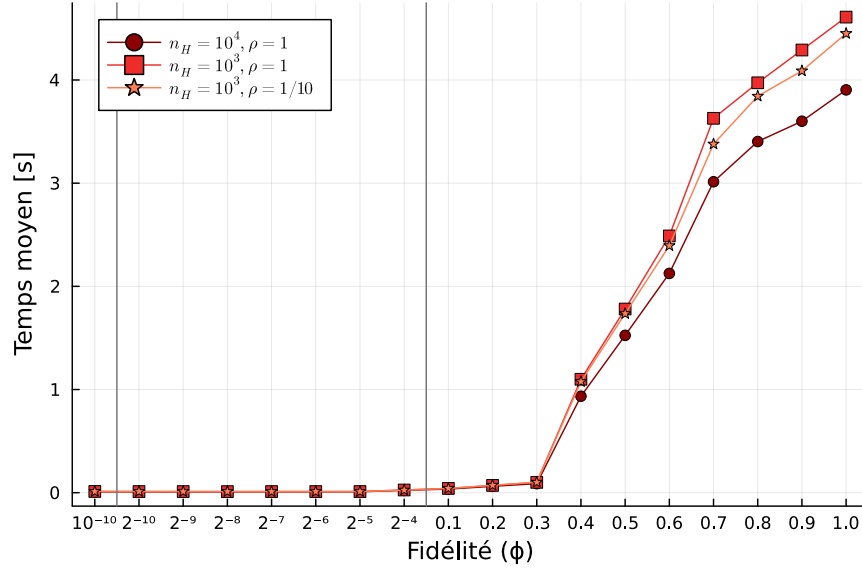


FIGURE 5.6 Valeurs des t_i de **solar3** avec différentes configurations d’hypercubes latins.

satisfaction, il est possible d’estimer qu’une optimisation basée sur matrice de biadjacence calculée à partir des analyses (a) et (b) risque de déterminer que tous les points évalués sont non réalisables. L’analyse (c) identifie avec plus de précision le comportement des contraintes en fonction de la fidélité. Aussi, tout comme pour **solar2**, les fidélités entre 2^{-10} et 2^{-4} n’apportent aucune information utile.

5.2.3 solar4

Rappelons que pour une configuration où $n_H = 10^3$, l’analyse n’est pas nécessairement effectuée sur 10^3 points, mais plutôt sur n_H^{ap} points, le nombre de points réalisables à priori. Pour les instances **solar2** et **solar3**, ce nombre de points n’a pas été discuté, car il est significativement grand pour toutes les configurations. Ce n’est toutefois pas le cas pour **solar4**, tel qu’indiqué au tableau 5.5.

TABEAU 5.5 Réalisabilité à priori des configurations de **solar4**.

Paramètre	Configuration		
ρ	1	1	1/20
n_H	10^4	10^3	10^3
n_H^{ap}	10	2	165

Il n'est donc pas surprenant que les figures 5.7 et 5.8 qui illustrent les proportions de représentativité et de satisfaction des contraintes montrent des résultats différents pour les différentes configurations. Les hypercubes latins effectués sur l'espace des paramètres entraînent en une analyse insuffisante. L'hypercube centré sur un point x^0 réalisable a échantillonné 165 points réalisables à priori, et l'analyse est davantage significative. Premièrement, les figures 5.7 et 5.8 révèlent que **solar4** comprend aussi plusieurs contraintes où la valeur de β ne change pas avec la fidélité pour le même point.

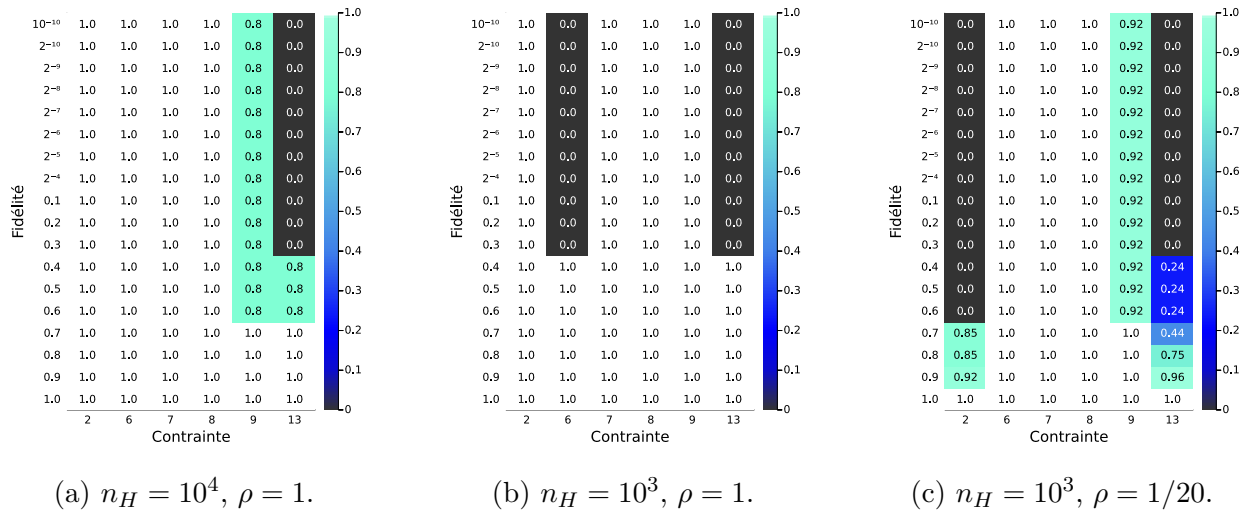


FIGURE 5.7 Valeurs des r_{ij} de **solar4** avec différentes configurations d'hypercubes latins.

Deuxièmement, le phénomène étudié avec **solar3** où pour plusieurs points, une fidélité n'est pas représentative malgré qu'elle identifie correctement si une contrainte est satisfaite à $\phi = 1$ survient également avec **solar4**, avec l'hypercube latin centré. Le phénomène est observé avec c_2 et c_{13} . Pour les points de l'échantillon, la contrainte c_2 se comporte exactement comme la contrainte c_2 de **solar3**. Or, la contrainte c_{13} est systématiquement satisfaite pour toute fidélité inférieure à $\phi = 0.3$, elle devient non satisfaite à $\phi = 0.4$, puis elle tend vers la vérité pour les fidélités supérieures. Il s'agit du comportement inverse à celui de la contrainte c_2 . Cela n'a toutefois pas causé de problèmes pour les analyses (a) et (b), car dans les deux cas, la proportion de satisfaction est nulle pour une fidélité supérieure à $\phi = 0.3$. Si ce n'était pas le cas, la situation observée avec c_2 est recréée, car l'analyse affirmerait que les basses fidélités sont représentatives, sans savoir que le phénomène en cause n'est pas le même pour les basses et les hautes fidélités. Le principe de la représentativité possède également de la valeur pratique pour ce cas, car il est inutile d'effectuer une sous-évaluation pour observer la

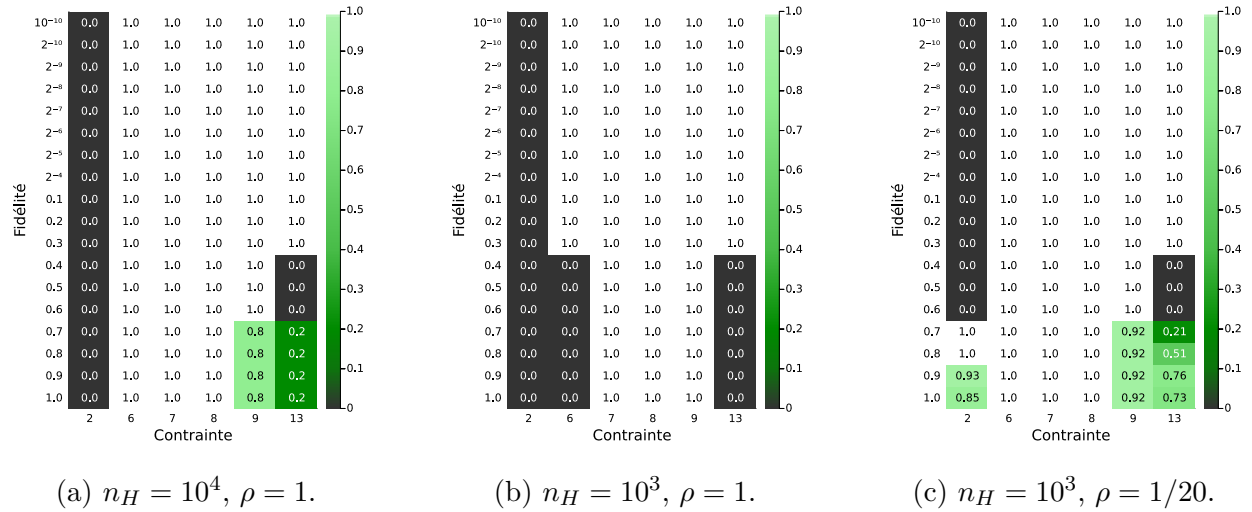


FIGURE 5.8 Valeurs des p_{ij} de **solar4** avec différentes configurations d'hypercubes latins.

valeur de la contrainte c_{13} pour une fidélité inférieure ou égale à 0.3 si le résultat est toujours que l'algorithme 3.2 passe à la sous-évaluation suivante. Une analyse plus complète comme celle en (c) pourrait permettre d'éviter ces évaluations supplémentaires.

Troisièmement, l'hypothèse (4.11) est satisfaite en (a) et en (b), mais ne l'est pas en (c) avec un écart maximal parmi les p_{ij} concernés de 0.03 avec $\varepsilon = 0.05$. Finalement, les temps moyens de **solar4** sont illustrés à la figure 5.9. Les relations entre les temps aux différentes fidélités ne sont pas les mêmes d'une configuration à l'autre, et il s'agit vraisemblablement d'une erreur statistique due aux deux échantillons de taille 2 et 10. Aussi, les fidélités entre 2^{-10} et 2^{-4} n'apportent aucune information utile. Pour **solar4**, il est certain que le choix de l'hypercube latin a un impact important sur l'optimisation qui en découlera, et cette boîte noire démontre l'importance de s'assurer que l'échantillon contient suffisamment de points réalisables a priori.

5.2.4 solar7

Pour **solar7**, les proportions r_{ij} et les proportions p_{ij} sont illustrées aux figures 5.10 et 5.11 respectivement. Les deux contraintes affectées par la multifidélité parmi les six contraintes du problème sont c_2 et c_6 . Il est observé que les trois configurations sont plutôt similaires. Avec $\varepsilon = 0.05$, les r_{ij} sont tels que $\phi_{i(2)} = \phi_{i(7)} = 2^{-4}$ en (a) et en (b). Ces fidélités sont toutes deux 10^{-10} en (c). Le point de départ choisi autour duquel les points sont échantillonnés pour l'hypercube latin centré se situe dans un sous-ensemble de l'espace des paramètres où les contraintes c_2 et c_6 sont rarement violées

Concernant les p_{ij} , l'hypothèse (4.11) n'est pas satisfaite avec des écarts maximaux parmi

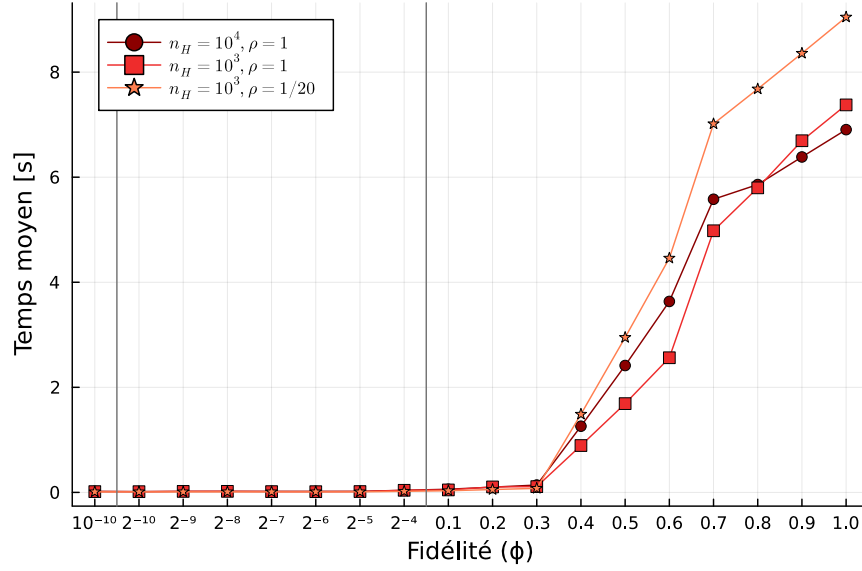
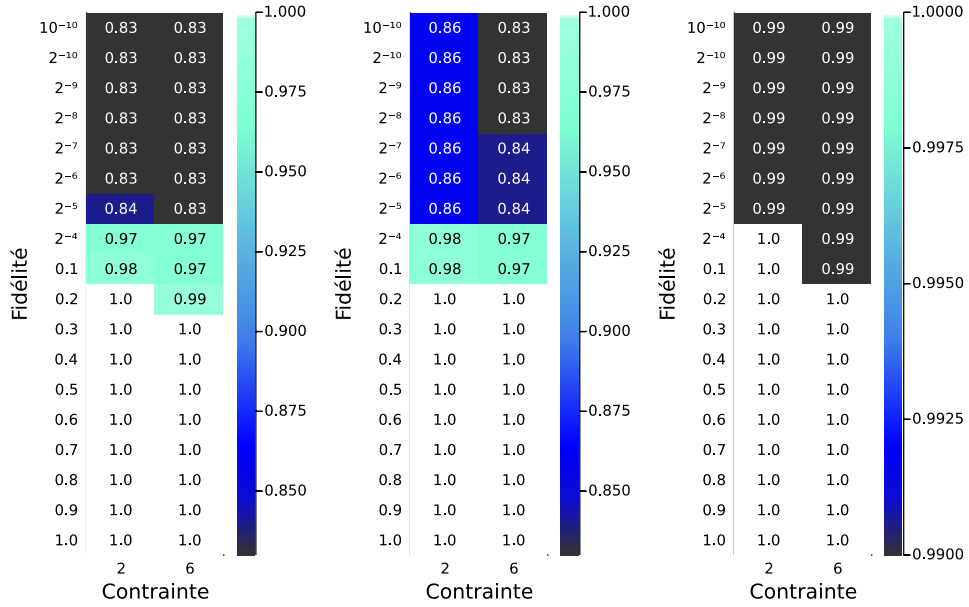


FIGURE 5.9 Valeurs des t_i de **solar4** avec différentes configurations d’hypercubes latins.



(a) $n_H = 10^4$, $\rho = 1$. (b) $n_H = 10^3$, $\rho = 1$. (c) $n_H = 10^3$, $\rho = 1/4$.

FIGURE 5.10 Valeurs des r_{ij} de **solar7** avec différentes configurations d’hypercubes latins.

les p_{ij} concernés de 0.03 en (a), de 0.03 en (b), et de 0.01 en (c) avec $\varepsilon = 0.05$. Encore une fois, ces écarts sont petits, le fait que l’hypothèse n’est pas respectée est donc peu problématique.

Les temps moyens pour **solar7** sont illustrés à la figure 5.12. Les relations entre les valeurs

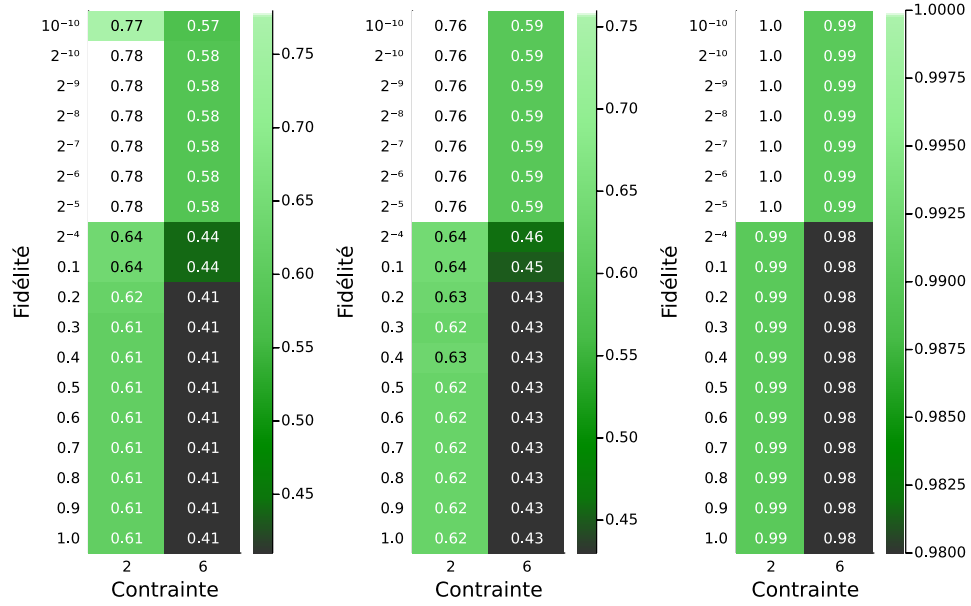


FIGURE 5.11 Valeurs des p_{ij} de *solar7* avec différentes configurations d'hypercubes latins.

de temps aux différentes fidélités sont similaires pour les différentes configurations. Le choix de la configuration pour *solar7* peut impacter les fidélités auxquelles sont assignées les contraintes c_2 et c_6 .

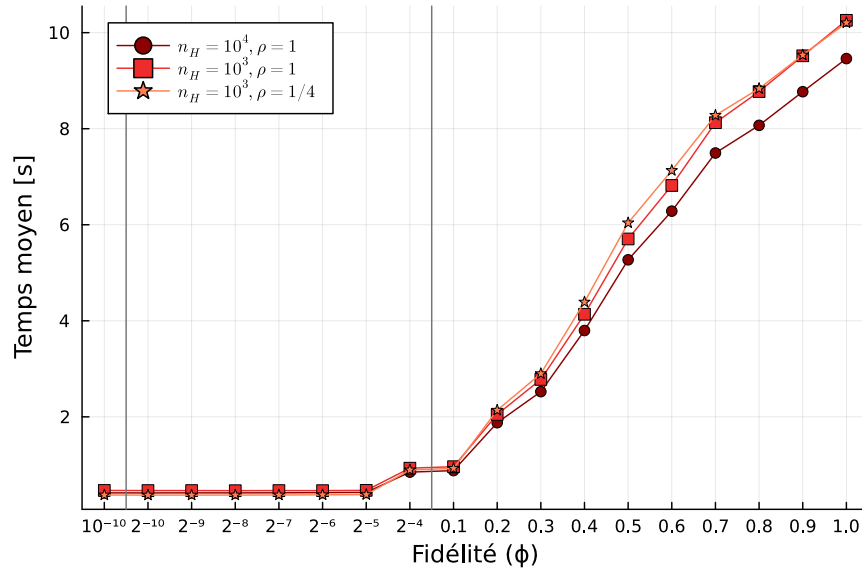


FIGURE 5.12 Valeurs des t_i de *solar7* avec différentes configurations d'hypercubes latins.

5.2.5 Conclusion sur les analyses du comportement des contraintes

Quatre conclusions principales ressortent des analyses sur les boîtes noires de la famille **solar**. Premièrement, les résultats pour les hypercubes latins de 10^3 points et ceux d'hypercubes latins de 10^4 points sont presque identiques. Il n'est donc pas nécessaire d'investir les ressources d'évaluer 10^4 points. Deuxièmement, il était inutile d'inclure autant de fidélités entre 10^{-10} et 0.1, celles-ci n'ont apporté aucune information pertinente pour les prochaines étapes de la méthode d'optimisation. Il est important de noter que ces deux conclusions sont propres aux boîtes noires **solar** étudiées, et peuvent différer pour d'autres problèmes. Troisièmement, après avoir choisi une valeur de n_H , il est important d'observer la valeur de n_H^{ap} après les évaluations, pour vérifier que l'échantillon considéré dans l'analyse est de taille significative. Dernièrement, lorsqu'une boîte noire possède des contraintes qui renvoient une valeur par défaut à certaines fidélités, il est possible que l'analyse soit insuffisante indépendamment de la taille de l'échantillon. Ce dernier doit contenir, pour chaque contrainte concernée, une proportion significative de points où la contrainte est satisfaite ou de points où elle n'est pas satisfaite par une marge pas trop grande, ainsi qu'une proportion significative de points où cette contrainte n'est pas satisfaite ou de points où elle est satisfaite par une marge pas trop grande. Il est à noter que cette propriété peut être observée dans un échantillon de points qui sont tous non réalisables ou tous réalisables, car elle a trait aux contraintes individuellement.

5.3 Calcul de la matrice de biadjacence

Suite aux analyses, la matrice de biadjacence optimale est calculée selon la seconde étape de l'algorithme 4.1 pour chaque configuration. On choisit une valeur de 0.05 pour le paramètre ε . Ayant observé à la section précédente que les fidélités correspondant à une puissance négative de 2 ne sont pas intéressantes, considérons plutôt l'ensemble

$$\Phi = \{10^{-10}, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}.$$

Lors de l'application de la méthode d'optimisation avec contraintes hiérarchisées, il n'est pas possible de manuellement modifier l'ensemble Φ . Or, ce n'est pas nécessaire, car la réduction 4.4.2 retire les fidélités qui n'apportent pas d'information utile. Ici, l'ensemble Φ est modifié pour alléger la présentation de la section. Dans la même optique, seulement les graphes réduits (comprenant les sommets de $\hat{I}$ et $\hat{J}$) sont présentés.

5.3.1 **solar2**

Débutons avec **solar2**, dont les assignations sont illustrées à la figure 5.13. Le problème (P) donné par les équations (4.1) à (4.5) comprend la variable $B \in \mathbb{B}^{\ell \times m}$. Comme **solar2** contient

13 contraintes et que Φ contient 11 fidélités, la matrice B contient 143 éléments. Ceux-ci étant binaires, la variable B peut prendre 2^{143} valeurs différentes. Après les quatre réductions, pour les trois configurations de **solar2**, seulement 1458 solutions restent. Il est évident que résoudre le problème (P_1) réduit en calculant la valeur de la fonction objectif pour 1458 solutions requiert un temps insignifiant pour tout ordinateur moderne en comparaison aux évaluations de **solar2** qui prennent en moyenne 29.1 secondes chacune selon le tableau 5.2.

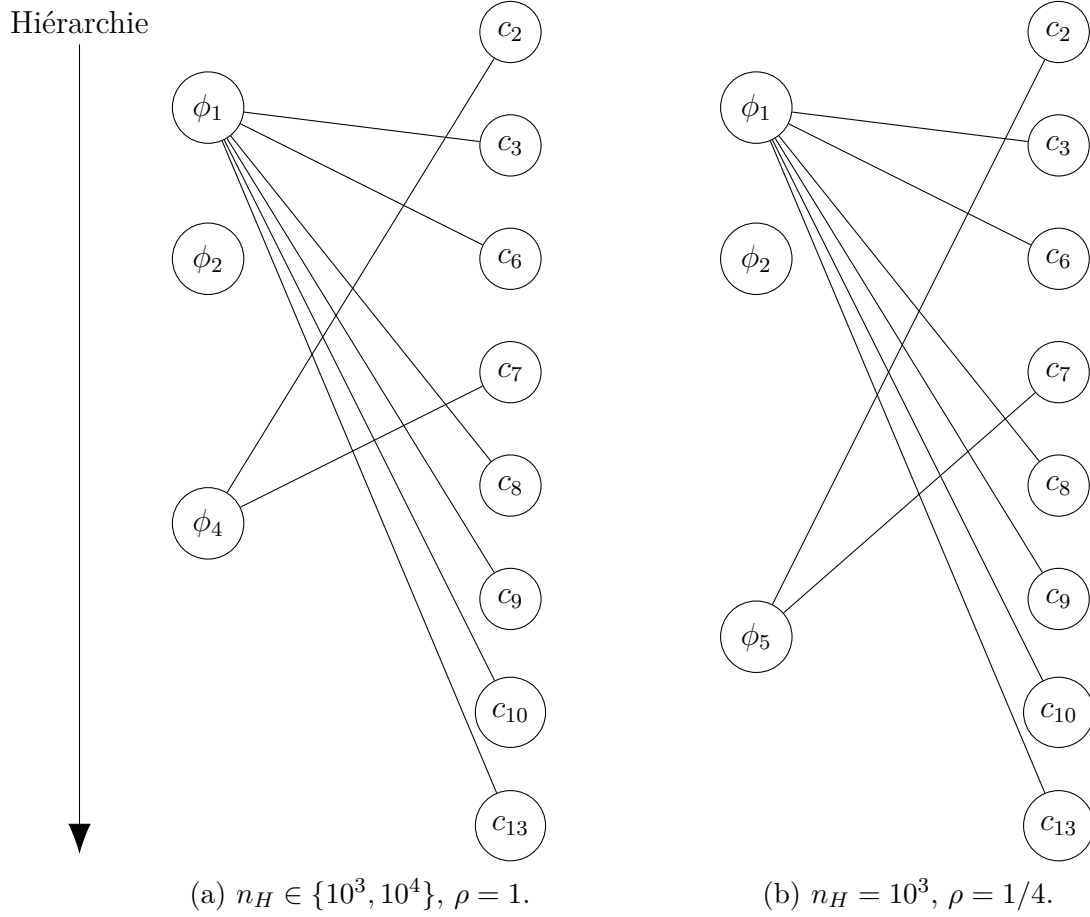


FIGURE 5.13 Graphes réduits représentant les assignations de **solar2**.

Le graphe résultant de la solution optimale est exactement le même pour les deux configurations de **solar2** qui échantillonnent l'espace des paramètres. Cela était attendu puisque les résultats des analyses sont presque identiques. Pour ces deux configurations, la réduction 4.4.2 a retenu les fidélités 0.3, 0.1 et 10^{-10} . Ces dernières correspondent aux indices $i(2)$, $i(7)$ et $i(j)$ pour toutes les autres contraintes c_j respectivement. Le modèle de matrice de biadjacence optimale a déterminé que pour les trois configurations, la sous-évaluation à ϕ_2 est une perte de temps; il est optimal d'assigner c_7 à une fidélité un peu plus élevée que $\phi_{i(7)}$,

où c_2 peut également être assignée. Toutes les autres contraintes peuvent être assignées à la fidélité 10^{-10} . Il est certain que lors de l'optimisation, peu de temps sera investi sur des points qui ne satisfont pas ces contraintes. L'unique différence avec l'hypercube latin centré est que la contrainte c_2 ne peut pas être assignée à une fidélité plus basse que ϕ_5 , c_2 et c_7 sont alors assignées à ϕ_5 .

5.3.2 solar3

En ce qui concerne **solar3**, dont les solutions optimales sont présentées à la figure 5.14, les valeurs de m et m_{ap} sont les mêmes que pour **solar2**. La variable B du problème (P) peut donc également prendre 2^{143} valeurs. Avec l'analyse provenant de l'hypercube latin centré, 1458 solutions restent après les réductions. Avec les deux autres analyses, ce nombre est de 128.

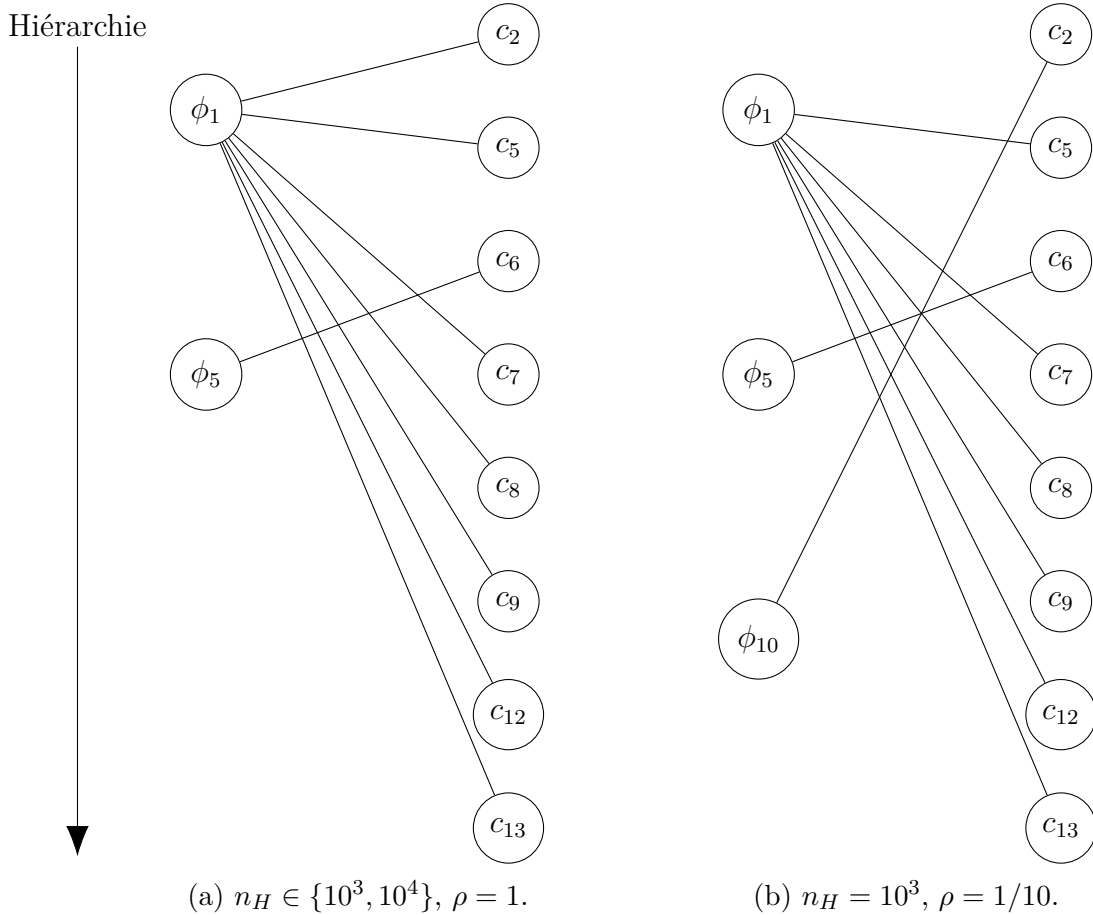


FIGURE 5.14 Graphes réduits représentant les assignations de **solar3**.

Les deux configurations où $\rho = 1$ ont trouvé la même solution optimale, ce qui s'explique

par les analyses similaires. La réduction 4.4.2 a retenu les fidélités 10^{-10} et 0.4. La contrainte c_6 ne peut être qu’assignée à $\phi_5 = 0.4$, et les 7 autres contraintes de $\hat{J}$ peuvent être assignées aux deux fidélités de $\hat{I}$, d’où les $2^7 = 128$ solutions. De toute évidence, la solution optimale est celle où ces 7 contraintes sont vérifiées à ϕ_1 , qui a un temps moyen d’évaluation t_1 d’environ 0.00798 seconde et d’environ 0.01206 seconde selon l’échantillon de 10^3 points et celui de 10^4 points respectivement. Avec l’hypercube latin centré, la fidélité ϕ_{10} est également retenue. Cela pose une situation semblable à celle de **solar2**, où à l’exception des contraintes qui peuvent être assignées à ϕ_1 , il y a une contrainte qui peut être assignée à l’une des deux autres fidélités, et la contrainte c_2 qui doit être assignée à la plus haute fidélité. Cela dit, la solution optimale pour **solar3** diffère de celle de **solar2**, car il est calculé que la sous-évaluation à ϕ_5 pour ne vérifier que c_6 (et revérifier les contraintes assignées à ϕ_1) a suffisamment de chances d’éviter le temps d’une sous-évaluation à la prochaine fidélité avec une interruption pour que le temps t_5 vaille l’investissement. En fin de compte, la seule différence entre les deux matrices de biadjacence est que la contrainte c_2 est assignée à la seconde plus haute fidélité en (b) au lieu de la plus basse fidélité en (a). Cette différence est importante; elle impactera vraisemblablement l’optimisation.

5.3.3 solar4

Pour l’instance **solar4**, les trois configurations ont donné trois matrices optimales différentes illustrées à la figure 5.15. La variable B du problème (P) possède 2^{176} valeurs possibles alors que la recherche exhaustive sur le problème (P_1) réduit est effectuée sur 128 solutions pour les échantillons avec $\rho = 1$, et 24 576 pour l’échantillon centré. Rappelons que le graphe trouvé en (a) est le résultat d’une analyse sur 2 points, et que le graphe trouvé en (b) est le résultat d’une analyse de 10 points. L’hypercube latin centré a entraîné des résultats plus fiables. Pour cette configuration, la réduction 4.4.2 a retenu les fidélités 1, 0.9, 0.7 et 10^{-10} , qui correspondent aux indices $i(2)$, $i(13)$, $i(9)$ et $i(j)$ pour les autres contraintes c_j respectivement. La solution optimale est celle où les contraintes qui doivent être assignées à une haute fidélité sont rassemblées à $\phi = 1$. Cette solution est significativement différente de celles trouvées par les deux autres configurations. Les sections 5.2.2 et 5.2.3 discutent de l’importance d’assigner c_2 à une haute fidélité, ce qui n’est pas le cas en (a) et en (b).

5.3.4 solar7

Pour terminer, le graphe pour chacune des trois configurations de **solar7** est présenté à la figure 5.16. Le sommet ϕ_{11} comporte un contour épais pour rappeler que cette sous-évaluation doit avoir lieu. Les deux hypercubes latins qui échantillonnent sur l’espace des paramètres ont la même solution, car les analyses sont similaires. Dans les deux cas, $\phi_{i(2)} = \phi_{i(6)} = 0.1$. Pour

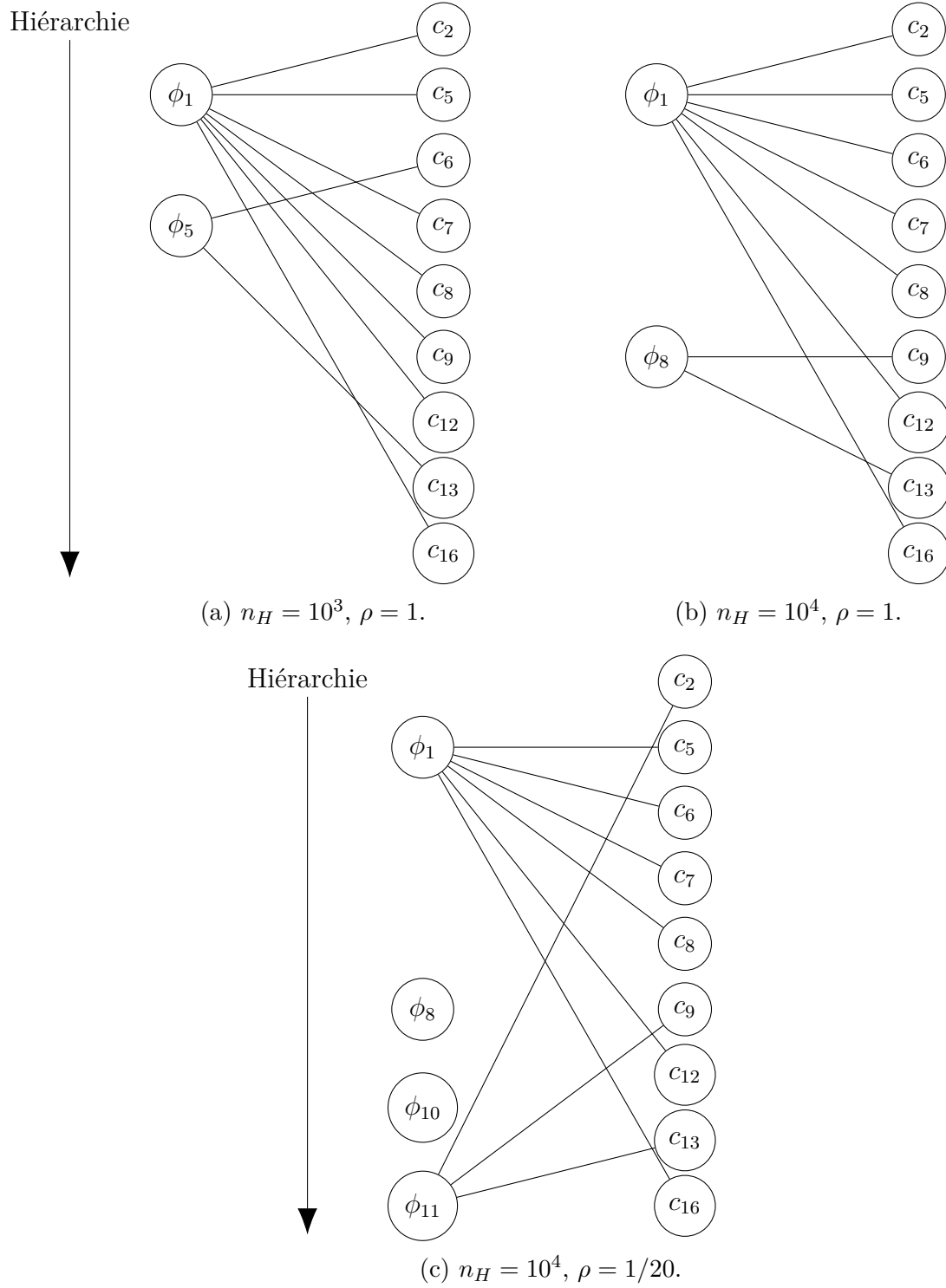


FIGURE 5.15 Graphes réduits représentant les assignations de solar4.

les autres instances de **solar**, toutes les contraintes qui ne sont pas affectées par la multifidélité

sont assignées à ϕ_1 . Ce n'est pas le cas pour **solar7**, la raison étant que la probabilité estimée d'effectuer une interruption à ϕ_1 avec ces contraintes est basse, et il est optimal de tout assigner à ϕ_2 , qui prend moins d'une demi-seconde de plus à évaluer en moyenne que ϕ_1 . À la figure 5.16 (a), la contrainte c_4 est plutôt assignée à ϕ_{11} . Cela s'explique par le fait que pour tous les points des deux échantillons, la contrainte c_4 est satisfaite à toute fidélité pour tous les points. Conséquemment, la solution présentée et la solution où toutes les contraintes sont assignées à ϕ_2 sont toutes deux optimales. Pour les deux configurations, le nombre de solutions pour B passe de 2^{66} à 32. Pour l'hypercube latin centré, cette quantité passe de 2^{66} à 15. La solution optimale avec l'analyse de cette configuration est de simplement effectuer une seule sous-évaluation à la fidélité $\phi = 1$.

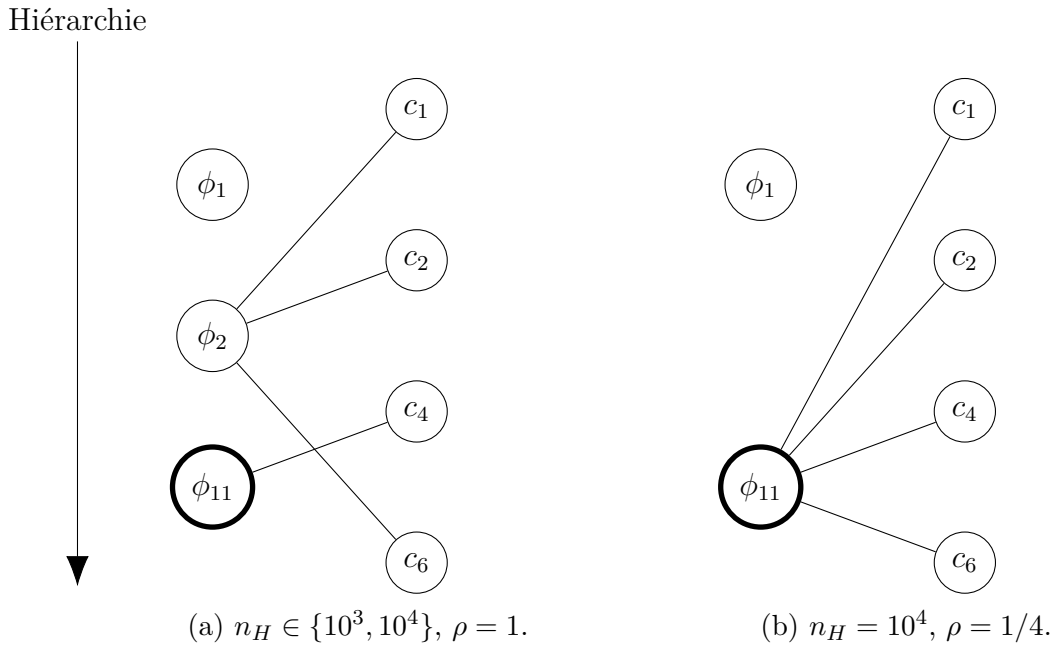


FIGURE 5.16 Graphes réduits représentant les assignations de **solar7**.

5.4 Conclusion sur les résultats intermédiaires

Pour conclure, l'observation de la section 5.2.5 que les hypercubes latins de 10^4 points ne sont pas nécessaires puisqu'un échantillon de taille 10^3 donne des résultats semblables est confirmée à la présente section. Les graphes résultant de ces échantillons sont non seulement semblables, mais ils sont identiques, à l'exception de **solar4** où la taille de n_H^{ap} est insuffisante. Cette dernière observation mène à la conclusion que le nombre de points n_H^{ap} doit être suffisamment grand, sans quoi l'analyse est peu pertinente. Dans le même ordre d'idées, ces

n_H^{ap} points doivent être variés en termes de satisfaction des contraintes. Aussi, les problèmes d'optimisation de matrice de biadjacence forment une catégorie de problèmes où des simplifications majeures sont possibles. Celles-ci permettent de réduire significativement la quantité de solutions possibles. Finalement, pour une application de la méthode d'optimisation avec contraintes hiérarchisées, la seule limitation dans le nombre de fidélités choisies n'est réellement que le temps d'évaluation de l'hypercube latin, car pour la suite de l'algorithme, les réductions retirent une quantité importante de fidélités non pertinentes.

CHAPITRE 6 RÉSULTATS DES OPTIMISATIONS

Ce chapitre présente les résultats d’optimisations effectuées en utilisant l’algorithme 4.1. Dans une première section, la pertinence du modèle d’optimisation d’une matrice de biadjacence est testée en comparant le modèle à d’autres méthodes pour déterminer cette matrice. Les trois sections qui suivent étudient l’efficacité de l’algorithme dans trois scénarios d’application différents. Pour terminer, une dernière section discute des résultats.

La méthode d’optimisation avec contraintes hiérarchisées doit être couplée à un solveur existant. Tous les tests sont effectués avec le solveur NOMAD 4, qui est présenté à la section 2.1. Ce solveur offre la possibilité d’utiliser différentes stratégies de gestion des contraintes, qui sont présentées à la section 2.2. En général, la BP donne de meilleurs résultats que la BE. Toutefois, la méthode testée est telle que les points non réalisables peuvent causer des interruptions, et les valeurs des contraintes renvoyées au solveur par l’algorithme 3.2 peuvent être différentes de la vérité. Le cas échéant, le calcul de $h(x)$ est imprécis. La BE est une manière d’indiquer au solveur de ne pas considérer ces valeurs en ignorant les points non réalisables. Différentes implémentations mettent en pratique cette idée.

Concernant le cas où le point de départ n’est pas réalisable, les implémentations de l’algorithme 4.1 sont divisées en deux phases consécutives.

- Phase de réalisabilité : phase visant à trouver un premier point réalisable.
- Phase d’optimalité : phase visant à trouver la meilleure solution réalisable possible.

Cette séparation en deux phases est utile, car l’algorithme 3.2 peut être nuisible durant la première phase. Durant celle-ci, à partir d’un point dans le domaine non réalisable, le solveur utilise les valeurs de $h(x)$ pour déplacer la solution vers le domaine réalisable. Or, si pour un point x évalué, une interruption a lieu et les valeurs des contraintes renvoyées au solveur sont significativement différentes de la vérité, la valeur de $h(x)$ ne permettra peut-être pas au solveur de trouver des directions vers le domaine réalisable. En revanche, durant la seconde phase, plusieurs points réalisables sont généralement évalués, et le solveur utilise $f(x)$ pour orienter les prochaines itérations. De ce fait, une implémentation en deux phases signifie qu’à la phase de réalisabilité, une seule sous-évaluation avec $\phi = 1$ est effectuée à chaque évaluation, et à la phase d’optimalité, l’algorithme 3.2 est lancé à chaque évaluation. Les implémentations testées sont les suivantes :

- Inter BP : une phase d’optimalité avec la BP où l’algorithme 3.2 est lancé à chaque évaluation ;
- Inter BE : une phase d’optimalité avec la BE où l’algorithme 3.2 est lancé à chaque évaluation ;

- Inter BP/BP : Implémentation en deux phases avec la BP ;
- Inter BE/BE : Implémentation en deux phases avec la BE ;
- Inter BP/BE : Implémentation en deux phases où la BP est utilisée pour la première, et la BE est utilisée à la seconde phase. Cette méthode est différente de la BPE présentée à la section 2.2.

Pour chaque optimisation, un graphique du nombre d'évaluations en fonction du temps pour les différentes implémentations testées est d'abord montré. Ce dernier permet de visualiser par quelle mesure chaque implémentation réduit le temps des évaluations. Les interruptions peuvent toutefois affecter la suite de points x^k . Des interruptions qui réduisent le temps d'évaluation n'impliquent pas nécessairement que l'optimisation se termine avec une meilleure solution. Ainsi, les graphes de convergence des différentes implémentations testées sont montrés à droite du premier graphique. Ceux-ci permettent de visualiser par quelle mesure chaque implémentation affecte la meilleure solution trouvée en fonction du temps. Dans ces graphiques, l'axe des abscisses représente le temps cumulé des évaluations de la boîte noire seulement. La raison de cette pratique est que la méthode d'optimisation avec contraintes hiérarchisées est développée dans le but d'être appliquée à des boîtes noires où le temps de calcul utilisé par le solveur (en l'occurrence NOMAD) et l'algorithme 3.2 est négligeable en comparaison au temps d'évaluation. La boîte noire de PRIAD est un tel problème. Or, les boîtes noires de la famille **solar** avec lesquelles les tests sont effectués sont plutôt rapides à évaluer.

6.1 Validation du calcul de la matrice de biadjacence

Pour valider l'approche présentée au chapitre 4, des optimisations avec l'algorithme 3.2 sont lancées avec des matrices de biadjacence B trouvées avec différentes stratégies, et ce, pour les instances **solar2**, **solar3** et **solar4** présentées au tableau 5.1. Comme le problème **solar7** est une boîte noire où il est attendu que la méthode d'optimisation avec contraintes hiérarchisées ne soit pas efficace, ce problème n'est pas inclus. Les stratégies de calcul de B sont les suivantes :

- Modèle $\varepsilon = 0.05$: Résolution du modèle de matrice de biadjacence optimale tel que montré au chapitre 4, avec $\varepsilon = 0.05$;
- Min temps $\varepsilon = 0$: Chaque contrainte c_j est assignée à la plus petite fidélité de Φ où la probabilité de représentativité estimée est de 1. Autrement dit, toute c_j est assignée à $\phi_{i(j)}$ avec $\varepsilon = 0$;
- Aléatoire $\varepsilon = 0$: Chaque contrainte est assignée à une fidélité aléatoire parmi trois fidélités de Φ choisies aléatoirement où la probabilité de représentativité estimée est de 1 ;

- Aléatoire total : Chaque contrainte est assignée à une fidélité aléatoire parmi trois fidélités de Φ choisies aléatoirement.

Toutes les stratégies sont basées sur l'ensemble de fidélités

$$\Phi = \{10^{-10}, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\},$$

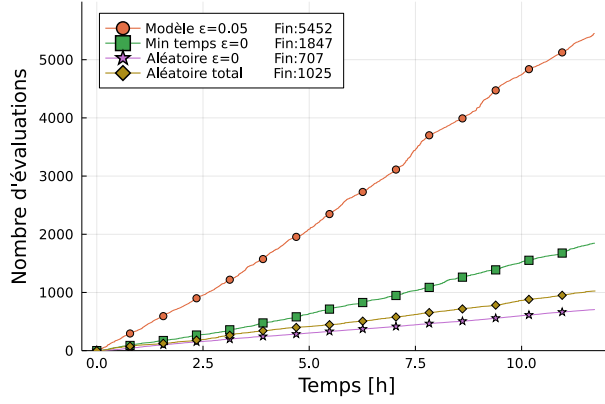
et les trois premières stratégies utilisent les analyses provenant des hypercubes latins centrés du chapitre 5, car ce sont eux qui donnent les analyses les plus complètes. Pour toutes les optimisations, le point de départ est le centre de l'hypercube latin concerné. Pour les deux dernières stratégies, trois fidélités sont choisies aléatoirement pour limiter le nombre de sous-évaluations. Sans cela, ces méthodes risquent de causer une grande quantité de sous-évaluations, ce qui leur cause un désavantage important.

6.1.1 solar2

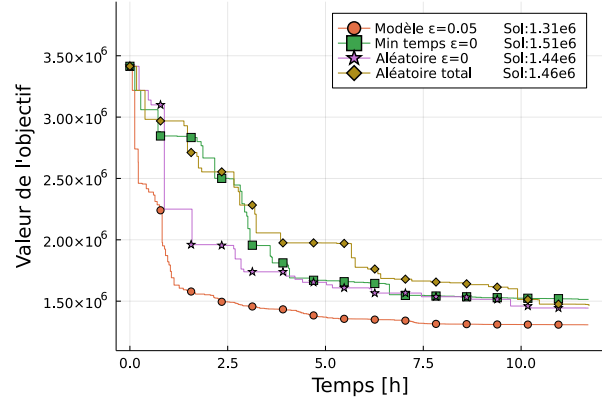
Avec l'instance **solar2**, les assignations données par la stratégie *Modèle* $\varepsilon = 0.05$ sont illustrées à la figure 5.13 (b). Pour les trois autres, les graphes réduits résultants sont illustrés à la figure A.1. Les quatre optimisations sont illustrées à la figure 6.1. Dans les boîtes de légende de ces figures, le terme « Sol » désigne la valeur de la meilleure solution obtenue, et le terme « Fin » désigne le nombre total d'évaluations. On y remarque qu'en tout point, la stratégie reposant sur le modèle de matrice de biadjacence optimale est celle qui a effectué plus d'évaluations, et qui a trouvé la meilleure solution. Aussi, malgré que la stratégie *Aléatoire* $\varepsilon = 0$ effectue le plus petit nombre d'évaluations, ce qui s'explique par les assignations aux hautes fidélités, cette stratégie est pour une grande partie du graphe de convergence la seconde meilleure. Cela démontre que les assignations ont un impact non seulement sur le temps, mais aussi sur les directions employées par la recherche directe.

6.1.2 solar3

Pour l'instance **solar3**, les assignations données par la première stratégie sont illustrées à la figure 5.14 (b). Pour les trois autres, les graphes réduits résultants sont illustrés à la figure A.2. La stratégie *Aléatoire total* ne trouve aucune solution réalisable, et ce, même si le point de départ est réalisable. Cela s'explique par les observations de la section 5.2.2 : la contrainte c_2 étant assignée à $\phi_7 = 0.6$, elle renvoie toujours une valeur de 100, et la dernière sous-évaluation ne renvoie jamais un point réalisable au solveur. Les trois autres stratégies ont évité ce problème en donnant à ε une valeur de 0.05 ou 0. Ces optimisations valident la pertinence de ce paramètre. Contrairement à **solar2**, la suite des points évalués est exactement la même pour chaque stratégie (à l'exception de *Aléatoire total*). Les courbes sont toutefois



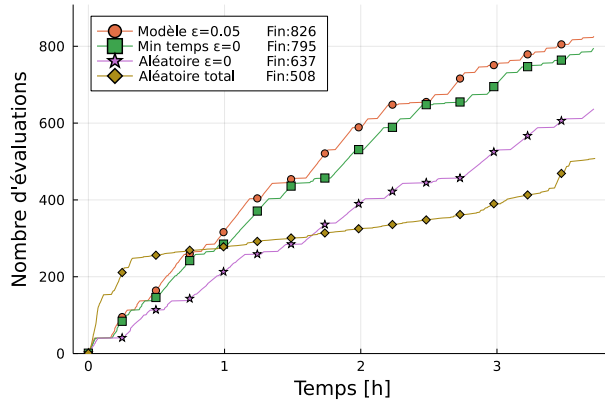
(a) Nombre d'évaluations en fonction du temps



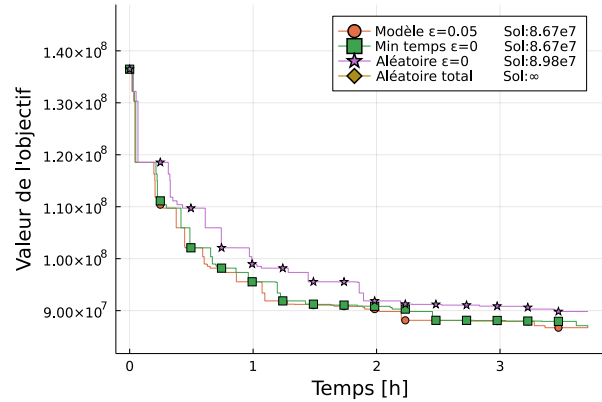
(b) Graphes de convergence

FIGURE 6.1 Optimisations de *solar2* avec différentes stratégies de calcul de B .

différentes car les points ne sont pas trouvés aux mêmes moments. Celles qui trouvent la meilleure solution sont donc celles qui effectuent le plus d'évaluations. En l'occurrence, il s'agit de *Modèle $\varepsilon=0.05$* pour tout temps, ainsi que de *Min temps $\varepsilon=0$* par moments.



(a) Nombre d'évaluations en fonction du temps



(b) Graphes de convergence

FIGURE 6.2 Optimisations de *solar3* avec différentes stratégies de calcul de B .

6.1.3 solar4

Pour terminer, avec *solar4*, les assignations données par la première stratégie sont illustrées à la figure 5.15 (c). Pour les trois autres, les graphes réduits résultants sont illustrés à la figure A.3. Comme pour *solar3*, *Aléatoire total* ne trouve aucune solution réalisable (le même phénomène étant en cause), la suite de points évalués est la même pour les trois premières stratégies, et la stratégie *Modèle $\varepsilon=0.05$* trouve la meilleure solution simplement parce qu'elle

effectue le plus grand nombre d'évaluations. La stratégie *Aléatoire total* a assigné la contrainte c_2 à la fidélité 0.3. On observe une seconde fois le phénomène survenu avec *solar3*, en plus du fait que NOMAD atteint un critère de convergence rapidement, car les évaluations à $\phi = 0.3$ sont rapides.

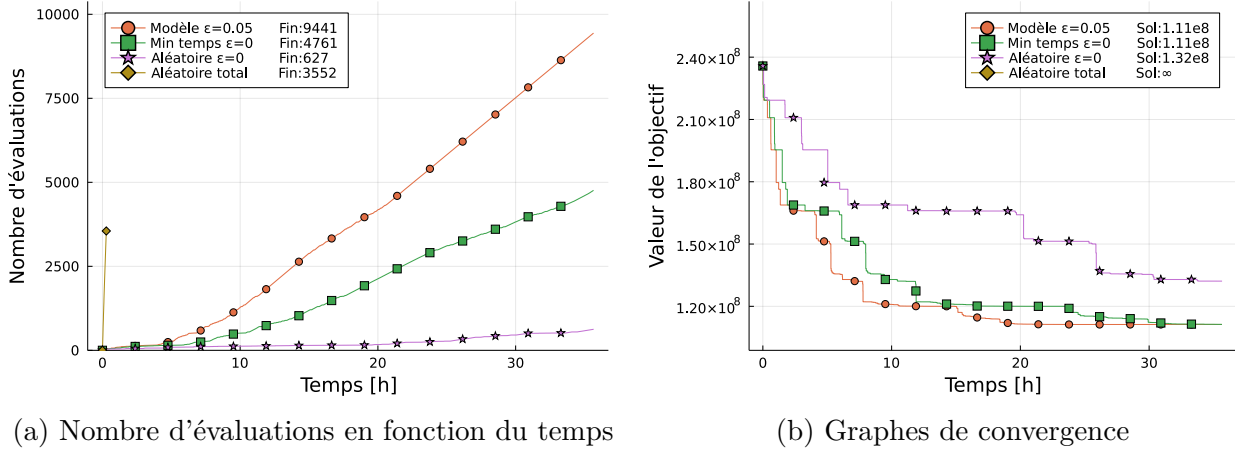


FIGURE 6.3 Optimisations de *solar4* avec différentes stratégies de calcul de B .

Pour conclure cette section, la méthode de calcul de la matrice de biadjacence présentée au chapitre 4 est valide, par l'observation qu'elle effectue une grande quantité d'évaluations qui aident à trouver des meilleures solutions plus rapidement que les autres stratégies. Aussi, la pertinence des interruptions dépend en partie de la valeur de ϵ . Plus elle est petite, plus les sous-évaluations sont effectuées à haute fidélité, donc plus elles sont coûteuses. Inversement, plus elle est grande, plus les interruptions sur des points réalisables sont fréquentes. Une valeur de 0,05 semble être un juste milieu.

6.2 Résultats numériques des optimisations

Cette section vise à comparer les implémentations de la méthode d'optimisation avec contraintes hiérarchisées à un cas de base dans le contexte de scénarios dans lesquels un utilisateur de la méthode pourrait se retrouver. Les optimisations présentées correspondent à la troisième étape de l'algorithme 4.1, et font suite aux sections 5.2 et 5.3, qui présentent respectivement les étapes un et deux de l'algorithme. Le cas de base auquel sont comparées les implémentations se décrit par le lancement de NOMAD avec les paramètres par défaut (en particulier, avec la BP) sur une boîte noire directement, en fixant la fidélité à 1.

6.2.1 Avec un point de départ non réalisable provenant d'un échantillon

Le scénario présenté par cette sous-section est celui où l'utilisateur lance l'algorithme 4.1 sans point de départ (tel que décrit à la section 4.5) avec $\rho = 1$ et $n_H = 10^3$, et où l'hypercube latin ne trouve pas de solution réalisable. Les tests sont effectués avec **solar2**, **solar3** et **solar4**, car ce sont les trois instances où la configuration d'échantillon $\rho = 1$ et $n_H = 10^3$ réplique ce scénario à la section 5.2. Pour le problème **solar2**, trois optimisations avec des points de départ différents sont illustrées aux figures 6.4, 6.5 et 6.6.

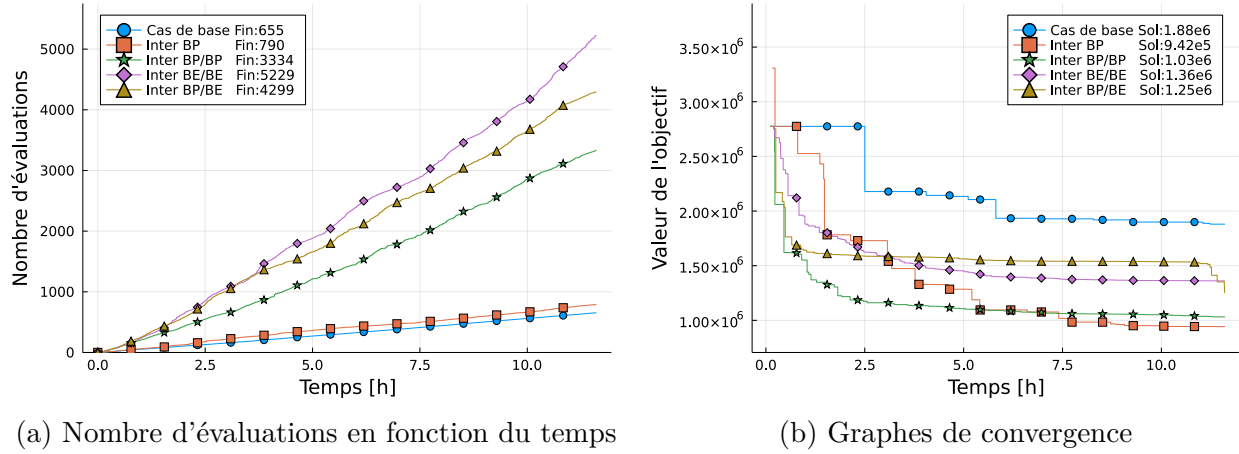


FIGURE 6.4 Optimisation de **solar2** à partir du meilleur point de départ non réalisable.

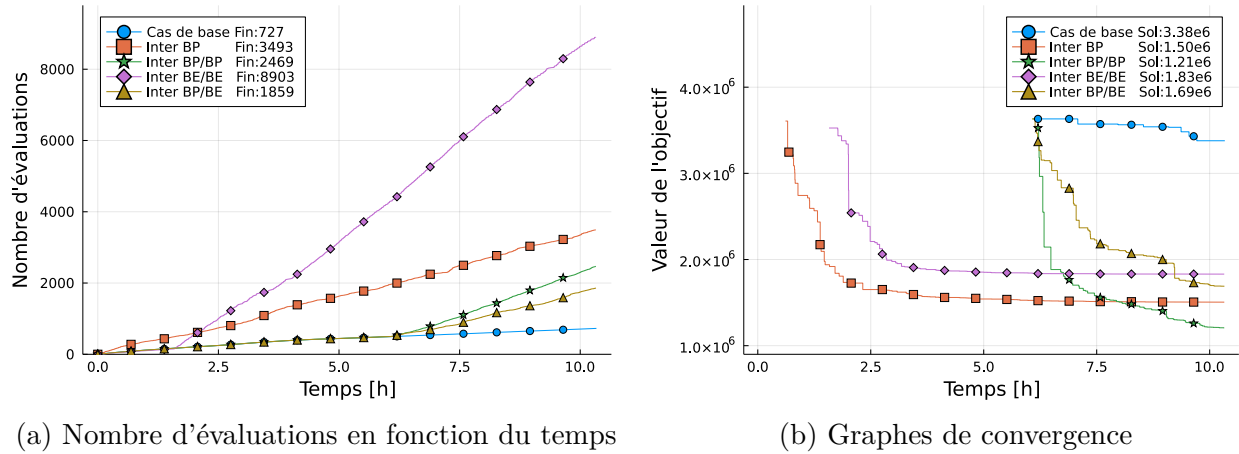


FIGURE 6.5 Optimisation de **solar2** à partir du 5e point de départ non réalisable.

La figure 6.4 montre l'optimisation à partir du point de l'hypercube latin qui minimise h , et les deux autres optimisations ont été sélectionnées parmi 20 avec des points de départ

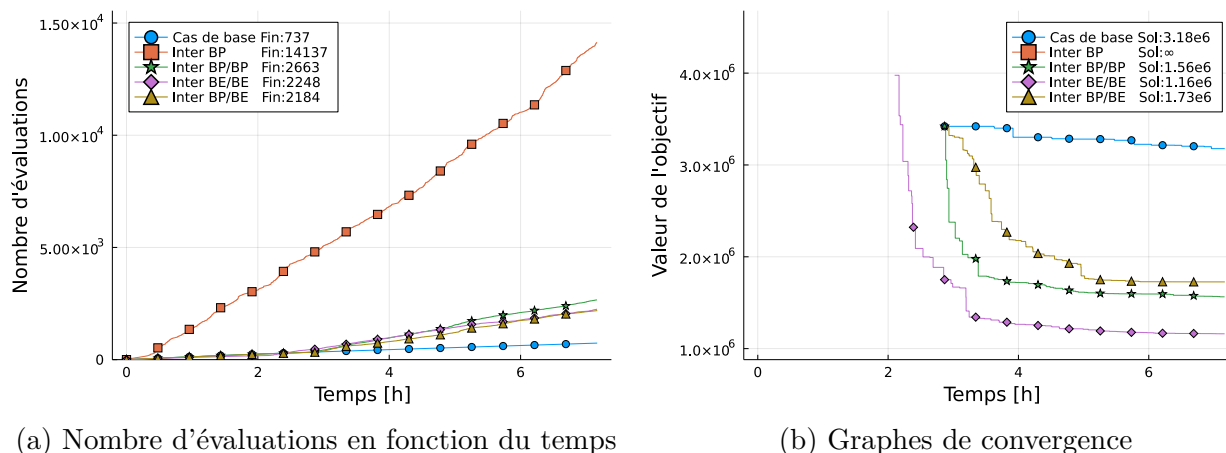


FIGURE 6.6 Optimisation de **solar2** à partir du 14e point de départ non réalisable.

différents pour montrer des comportements particuliers. Pour toutes ces optimisations, le cas de base est celui qui effectue le moins d'évaluations et qui trouve les moins bonnes solutions. La figure 6.5 illustre un cas où une solution réalisable trouvée tardivement donne tout de même lieu à de bonnes solutions durant la phase d'optimalité. Un cas fréquent est celui montré par la figure 6.6, où l'implémentation qui ne comprend pas de phase de réalisabilité effectue un grand nombre d'évaluations rapides sans succès. Le tableau 6.1 montre dans la colonne de gauche combien d'optimisations sur les 20 effectuées ont trouvé un point réalisable, et dans la colonne de droite combien ont trouvé un point réalisable alors que le cas de base n'en a pas trouvé, et ce, pour chaque implémentation.

TABLEAU 6.1 Résultats de réalisabilité de 20 optimisations de **solar2** à partir de points de départ non réalisables.

Implémentation	Nb. réalisable (%)	Nb. réalisable et base NR (%)
Cas de base	10 (50)	0 (0)
Inter BP	3 (15)	0 (0)
Inter BP/BP	10 (50)	0 (0)
Inter BE/BE	10 (50)	3 (15)
Inter BP/BE	10 (50)	0 (0)

On constate que l'implémentation *Inter BP* n'est pas efficace pour trouver une première

solution réalisable à partir d'un point de départ non réalisable. Aussi, lorsque la BE est employée à la phase de réalisabilité, une solution réalisable est trouvée alors que la BP n'en a pas trouvée à trois occasions. D'un autre côté, il y a également trois points de départ à partir desquels la BP a trouvé une solution réalisable, mais pas la BE. Les résultats des 20 optimisations sont illustrés par les profils de données de la figure 6.7. Au profil (a), le plus grand τ qui permet de discerner les implémentations à deux phases est utilisé. Ce τ vaut 10^{-4} , ce qui implique qu'aucune de ces trois implémentations ne démarque de manière significative. Au profil (b), malgré que la précision soit basse ($\tau = 0.5$), les taux de résolution des problèmes sont plutôt bas. Cela s'explique par le fait que plusieurs implémentations n'ont pas trouvé de solution réalisable pour un grand nombre d'optimisations.

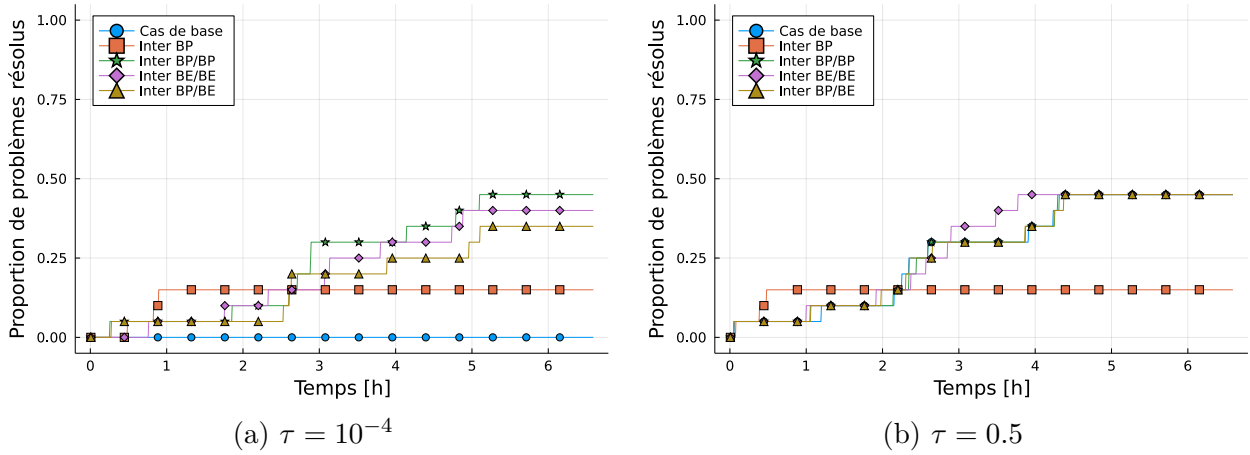


FIGURE 6.7 Profils de données de 20 optimisations de *solar2* à partir de points non réalisables.

En ce qui concerne les instances *solar3* et *solar4*, 20 optimisations ont également été effectuées, et les résultats de réalisabilité sont présentés aux tableaux 6.2 et 6.3. Les conclusions sont similaires à celles tirées pour *solar2* en ce qui a trait à la phase de réalisabilité. Toutefois, lors de la phase d'optimalité, pour les raisons expliquées à la section 5.2, l'algorithme 3.2 ne renvoie que des solutions non réalisables au solveur. On a donc que pour toutes les implémentations, au plus un point réalisable est trouvé par optimisation, alors que lorsque le cas de base en trouve un, il en trouve plusieurs autres meilleurs par la suite au cours de l'optimisation. Il est inutile de montrer des graphes de convergence ou des profils de données pour les instances *solar3* et *solar4*; le cas de base est évidemment préférable dans tous les résultats.

TABLEAU 6.2 Résultats de réalisabilité de 20 optimisations de **solar3** à partir de points de départ non réalisables.

Implémentation	Nb. réalisable (%)	Nb. réalisable et base NR (%)
Cas de base	6 (30)	0 (0)
Inter BP	0 (0)	0 (0)
Inter BP/BP	16 (30)	0 (0)
Inter BE/BE	16 (30)	1 (5)
Inter BP/BE	16 (30)	0 (0)

TABLEAU 6.3 Résultats de réalisabilité de 20 optimisations de **solar4** à partir de points de départ non réalisables.

Implémentation	Nb. réalisable (%)	Nb. réalisable et base NR (%)
Cas de base	2 (20)	0 (0)
Inter BP	0 (0)	0 (0)
Inter BP/BP	2 (20)	0 (0)
Inter BE/BE	3 (25)	1 (5)
Inter BP/BE	2 (20)	0 (0)

6.2.2 Avec un point de départ réalisable provenant d'un échantillon

Le scénario présenté par cette sous-section est celui où l'utilisateur lance l'algorithme 4.1 sans point de départ (tel que décrit à la section 4.5) avec $\rho = 1$ et $n_H = 10^4$, et où l'hypercube latin trouve une solution réalisable. Les tests sont effectués avec **solar2**, **solar3** et **solar7**, car ce sont les trois instances où la configuration d'échantillon $\rho = 1$ et $n_H = 10^4$ réplique ce scénario à la sous-section 5.2. La figure 6.8 présente une optimisation de **solar2** et la figure 6.9 illustre deux profils de données de 20 optimisations, toutes à partir de points de départ réalisables différents.

Sachant que la matrice de biadjacence est la même pour l'échantillon avec $n_H = 10^3$ points et l'échantillon de $n_H = 10^4$ points, les résultats des optimisations sont plutôt semblables à ceux de la section 6.2.1 en considérant *Inter BP/BP* et *Inter BP/BE* à partir du premier point réalisable trouvé. Une différence notable apparaît dans les profils de données de la figure 6.9, où la BE se démarque en (a). Encore une fois, avec **solar3**, aucun point réalisable n'est trouvé par la méthode d'optimisation avec contraintes hiérarchisées et le cas de base

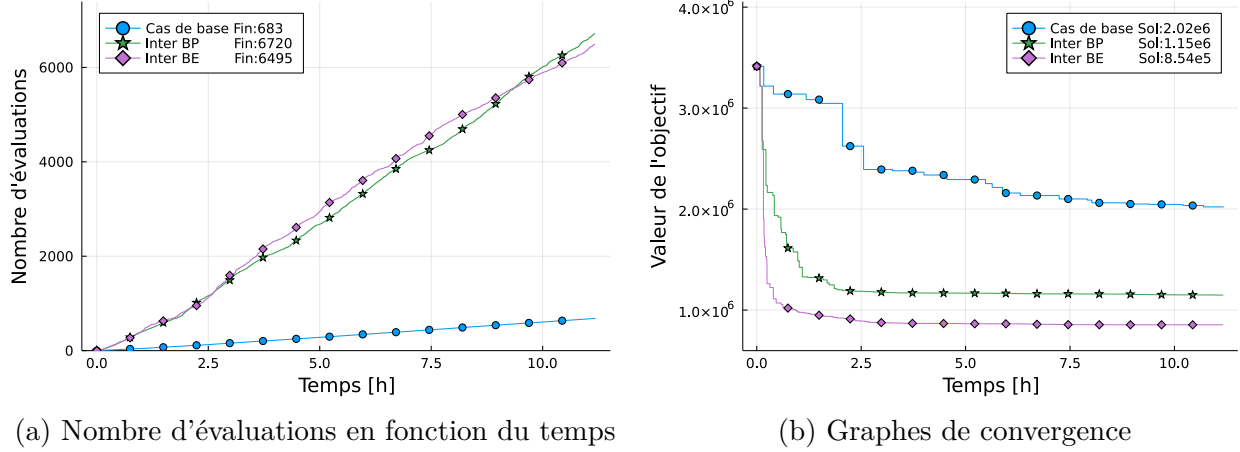


FIGURE 6.8 Optimisation de **solar2** à partir d'un point de départ réalisable.

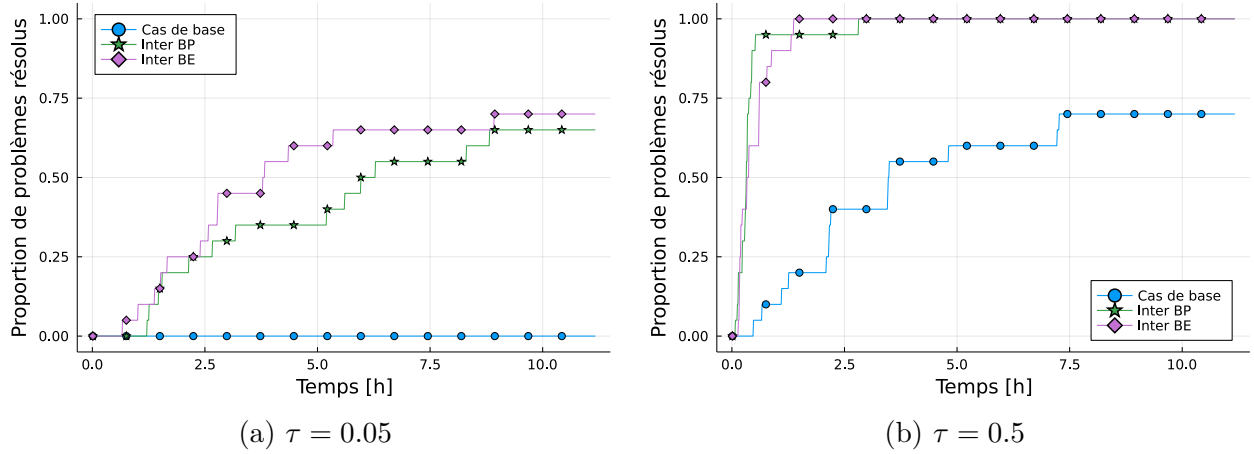
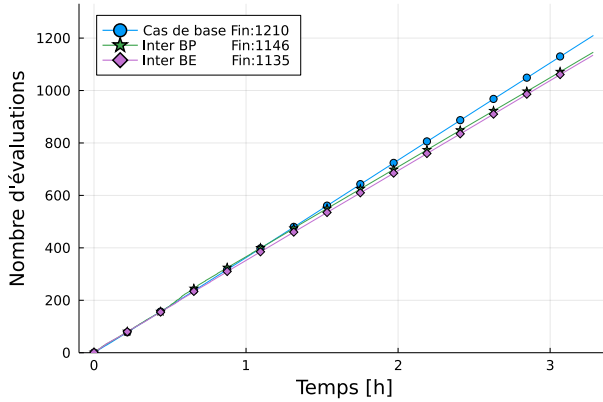


FIGURE 6.9 Profils de données de 20 optimisations de **solar2** à partir de points réalisables.

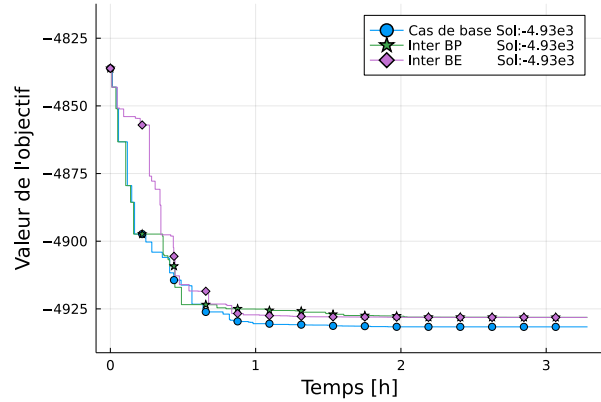
est préférable.

Au sujet de **solar7**, une optimisation est illustrée à la figure 6.10, et la figure 6.11 illustre deux profils de données de 20 optimisations, toutes à partir de points de départ réalisables différents. On constate que comme les contraintes de la boîte noire sont faciles à satisfaire, peu d'interruptions ont lieu. À la figure 6.10 (a), le cas de base effectue même plus d'évaluations, signifiant que les sous-évaluations supplémentaires à celle où $\phi = 1$ ajoutent plus de temps qu'elles en sauvent en trouvant des contraintes non satisfaites. Il en résulte que la solution du cas de base est légèrement meilleure aux autres. Cette observation est reflétée dans les profils de données, où malgré que les implémentations de la méthode d'optimisation avec contraintes hiérarchisées ont de meilleures solutions au début de l'optimisation, le cas de base termine

souvent avec une meilleure solution.

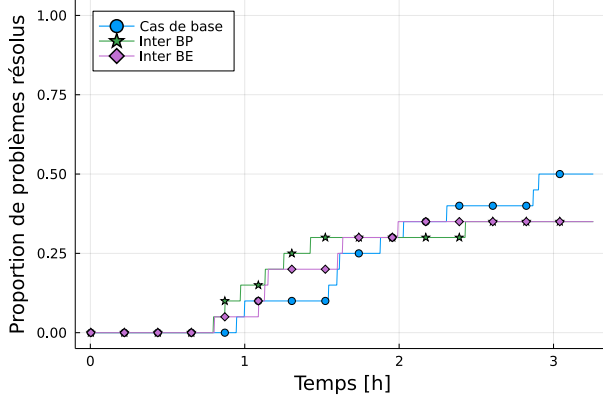


(a) Nombre d'évaluations en fonction du temps

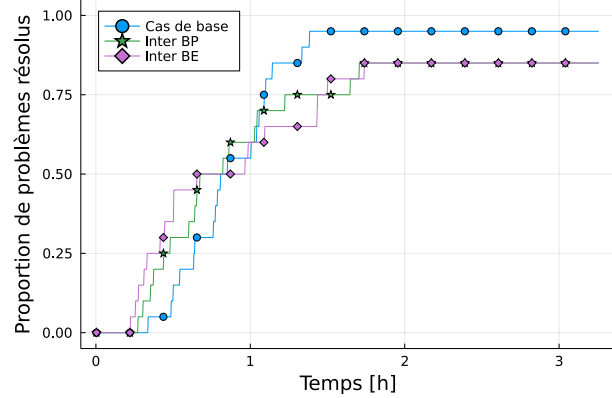


(b) Graphes de convergence

FIGURE 6.10 Optimisation de *solar7* à partir d'un point de départ réalisable.



(a) $\tau = 0.001$



(b) $\tau = 0.01$

FIGURE 6.11 Profils de données de 20 optimisations de *solar7* à partir de points réalisables.

6.2.3 Avec un point de départ réalisable connu

Le scénario présenté par cette sous-section est celui où l'utilisateur lance l'algorithme 4.1 avec un point de départ x^0 réalisable connu, avec $n_H = 10^3$, et où ρ vaut $1/4$, $1/10$, $1/20$ et $1/4$ pour *solar2*, *solar3*, *solar4* et *solar7*, respectivement. Pour ce scénario, le cas de base correspond à lancer directement l'optimisation sur le point de départ connu, alors que la méthode d'optimisation avec contraintes hiérarchisées doit évaluer les points d'un hypercube latin d'abord. De ce fait, le temps de ces évaluations est ajouté au temps d'optimisation.

Le projet étant en collaboration avec Hydro-Québec, les évaluations des hypercubes latins centrés sont effectuées avec la grappe de calcul de l'IREQ, où 640 cœurs ont été utilisés pour paralléliser les évaluations des hypercubes latins centrés. Les temps des évaluations sont de 642.22 secondes, 45.64 secondes, 94.75 secondes, et 181.07 secondes pour **solar2**, **solar3**, **solar4** et **solar7** respectivement. Il est important de rappeler que chaque point à échantillonner correspond à une évaluation par fidélité dans Φ .

La figure 6.12 présente une optimisation de **solar2**. En (a), les implémentations *Inter BP* et *Inter BE* effectuent respectivement environ 6.90 et 7.57 fois plus d'évaluations que le cas de base, et ces implémentations trouvent des solutions significativement meilleures en (b). Cela est reflété dans le profil de données de la figure 6.13 (a), où la méthode d'optimisation avec contraintes hiérarchisées résout une grande proportion de problèmes que le cas de base ne résout pas avec $\tau = 0.1$, et ce, peu importe la barrière utilisée. Avec une précision plus basse ($\tau = 0.5$), le profil de données de la figure 6.13 (b) montre que le cas de base dépasse *Inter BE*, car il y a une optimisation où *Inter BE* est grandement inefficace. En effet, pour cette optimisation, *Inter BE* n'améliore que de 0.48×10^6 la solution initiale, alors que *Inter BP* et le cas de base l'améliore de 2.22×10^6 et 2.07×10^6 , respectivement. Les profils de données sont le résultat de 20 optimisations, toutes avec une graine aléatoire différente pour le solveur NOMAD. Dans tous les cas, la BP est préférable.

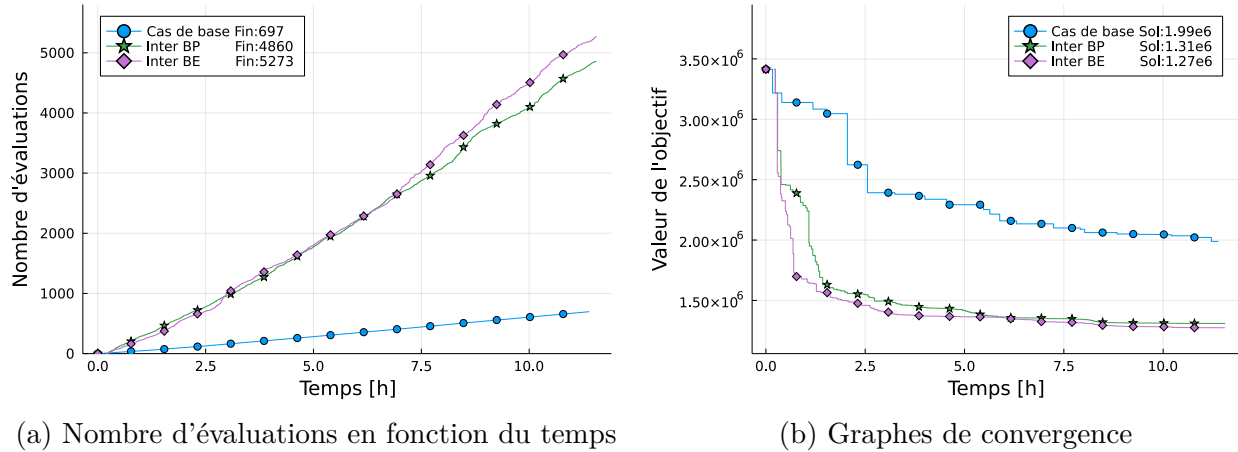


FIGURE 6.12 Optimisation de **solar2** à partir d'un point de départ réalisable connu avant l'optimisation.

Avec l'instance **solar3**, un résultat différent est observé. La figure 6.14 illustre une optimisation où l'implémentation *Inter BE* effectue environ 6.83 fois plus d'évaluations que les deux autres combinées, et où *Inter BP* effectue environ le même nombre d'évaluations que le cas de base. Ces deux courbes se suivent dans le graphe de convergence en (b), jusqu'à un

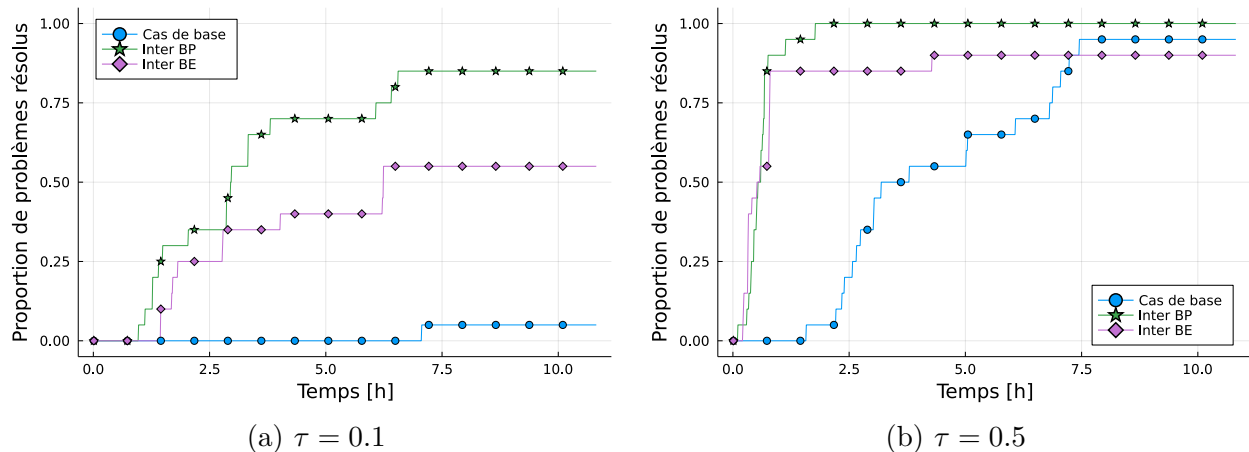


FIGURE 6.13 Profils de données de 20 optimisations de **solar2** avec un hypercube latin centré sur un point de départ réalisable.

moment où le cas de base trouve de meilleures solutions. La majorité des 20 optimisations effectuées avec différentes graines aléatoires pour NOMAD se comportent ainsi. La figure 6.15 illustre deux profils de données résumant ces 20 optimisations. L'implémentation *Inter BE* est l'implémentation la plus efficace par une grande marge, et les deux autres sont plutôt semblables.

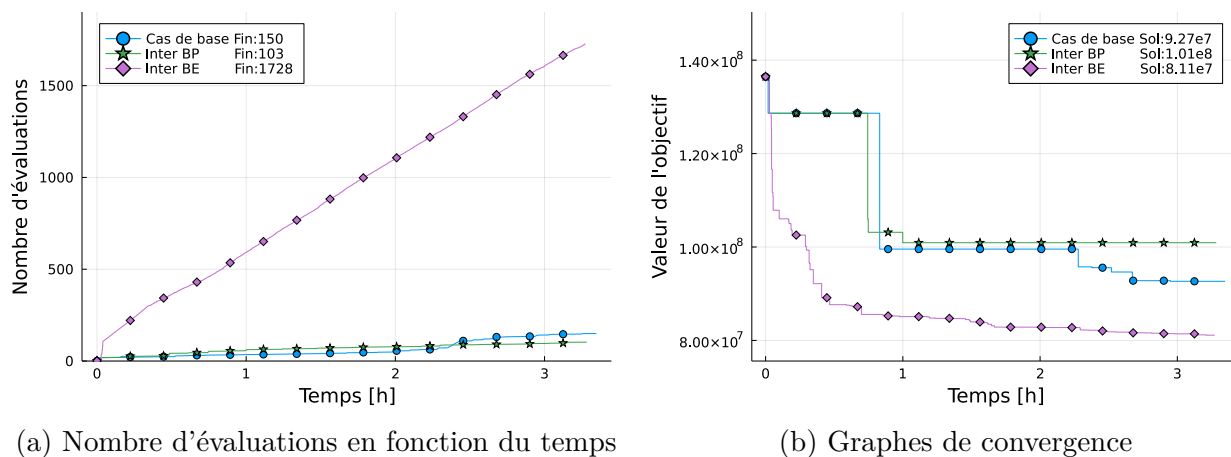


FIGURE 6.14 Optimisation de **solar3** à partir d'un point de départ réalisable connu avant l'optimisation.

Pour **solar4**, la figure 6.16 illustre une optimisation où un phénomène intéressant se produit. Durant la première moitié de l'optimisation, la recherche directe du cas de base évalue plusieurs points non réalisables particulièrement longs à évaluer. Ce phénomène est apparent

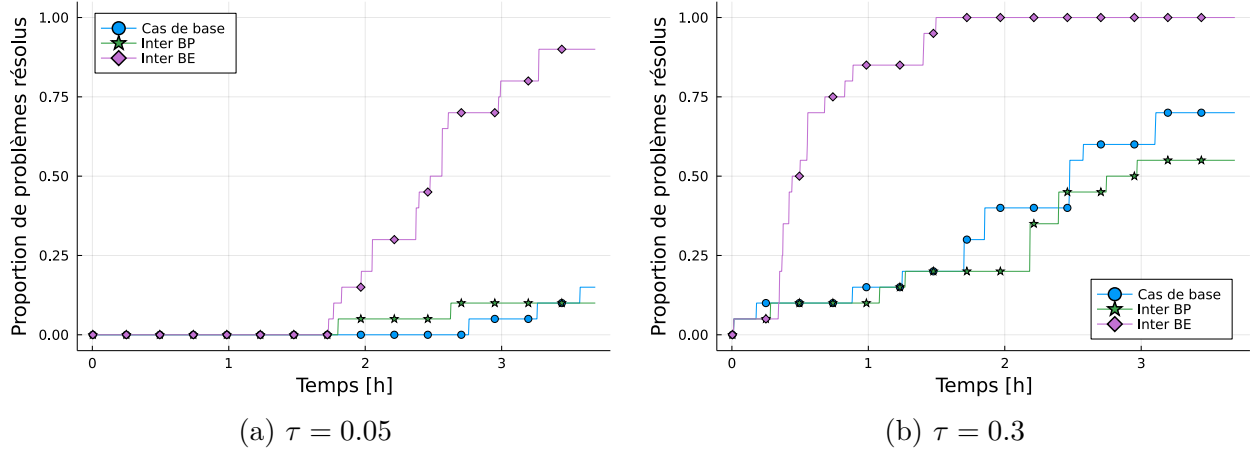


FIGURE 6.15 Profils de données de 20 optimisations de *solar3* avec un hypercube latin centré sur un point de départ réalisable.

en (a). Les implémentations *Inter BP* et *Inter BE* effectuent des interruptions rapides sur ces points, ce qui leur permet de trouver de meilleures solutions en moins de temps. Lorsque le cas de base n'évalue pas de points dans cette région de l'espace (avec d'autres graines aléatoires), les trois courbes sont plutôt semblables en termes de nombre d'évaluations et de graphe de convergence. En considérant les 20 optimisations avec différentes graines aléatoires, il est observé que l'implémentation *Inter BE* est souvent la meilleure, et que les deux autres varient d'une optimisation à l'autre. La figure 6.17 illustre deux profils de données de ces 20 optimisations.

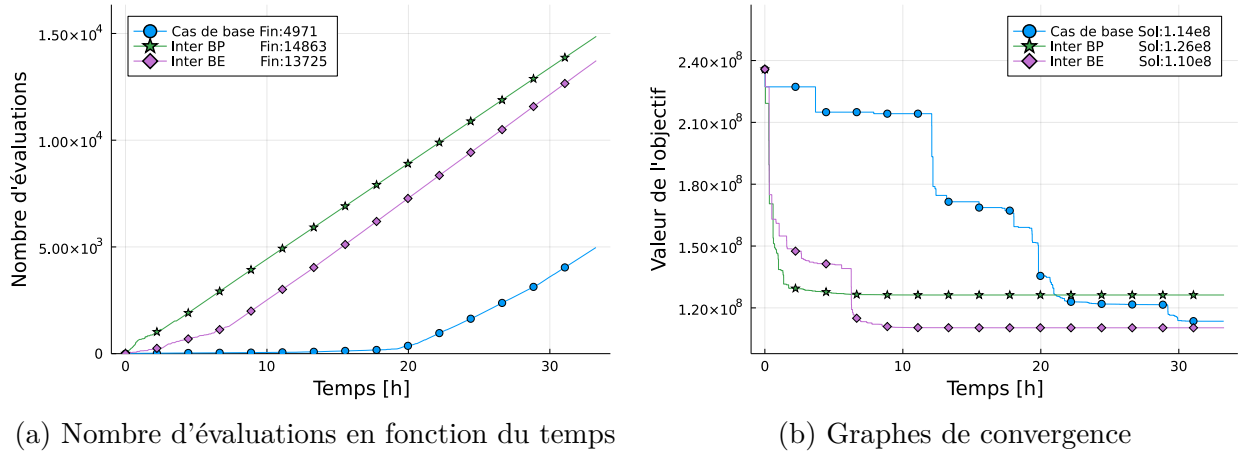


FIGURE 6.16 Optimisation de *solar4* à partir d'un point de départ réalisable connu avant l'optimisation.

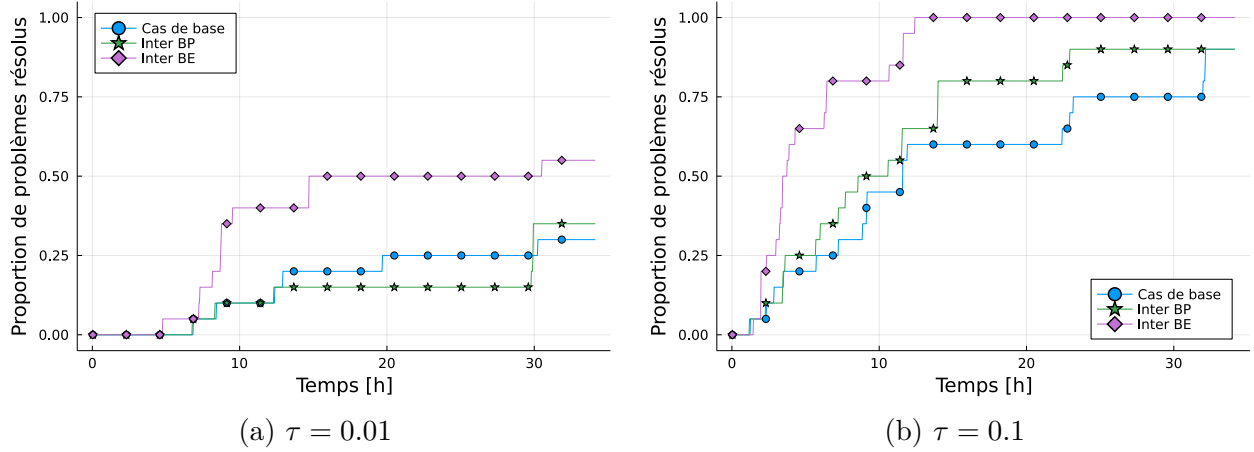


FIGURE 6.17 Profils de données de 20 optimisations de **solar4** avec un hypercube latin centré sur un point de départ réalisable.

En ce qui concerne **solar7**, comme les assignations sont équivalentes au cas de base (voir figure 5.16), les optimisations ne sont pas montrées. Celles-ci sont toutes identiques, à l'exception du cas de base qui a une avance constante de 181.07 secondes, ce qui est plutôt insignifiant en comparaison à un temps d'optimisation de plusieurs heures.

6.3 Discussion

D'abord, lorsque l'optimisation doit être lancée à partir d'un point non réalisable, les implémentations à deux phases trouvent significativement plus souvent des solutions réalisables que celles sans phase de réalisabilité. Cette dernière est donc nécessaire. En ce qui concerne le choix des barrières, les résultats actuels ne permettent que de conclure que la BP pour les deux phases est préférable pour **solar2**, et ce, seulement lorsqu'une haute précision sur la solution est demandée.

Considérant les trois scénarios étudiés, celui où un point de départ réalisable est connu à l'avance est celui où la méthode d'optimisation avec contraintes hiérarchisées est la plus efficace en comparaison au cas de base. Cela s'explique par deux phénomènes principaux. Premièrement, en échantillonnant plus près d'un point réalisable, la quantité de points n_H^{ap} risque d'être élevée. Considérant que la méthode vise à être appliquée avec des boîtes noires où les contraintes sont rarement satisfaites, il est hautement probable qu'un hypercube latin avec $\rho = 1$ trouve peu de points réalisables à priori avec ces boîtes noires. Deuxièmement, le comportement des contraintes en fonction de la fidélité n'est pas nécessairement le même à différents endroits de l'espace des paramètres. Lorsque ce n'est pas le cas, comme avec **solar2**, la méthode est efficace, peu importe le scénario. D'un autre côté, **solar3** et **solar4** ont montré

que lorsque des contraintes ont certaines propriétés, un échantillonnage local est nécessaire. Concernant les différentes implémentations de la méthode, la meilleure stratégie de gestion des contraintes dépend de la boîte noire optimisée, mais les résultats indiquent que la BE est le plus souvent la plus efficace.

La réussite ou l'échec de la méthode d'optimisation par contraintes hiérarchisées dépend pour la plus grande part de la qualité de l'analyse, et cette dernière est la plus élevée dans le scénario où l'on connaît un point réalisable avant l'optimisation et en choisissant une valeur de ρ inférieure à 1. Étant donné une telle analyse de haute qualité, la méthode résout avec succès des problèmes d'optimisation avec des contraintes difficiles à satisfaire (**solar2** et **solar3**) ainsi que des problèmes d'optimisation où les points réalisables sont particulièrement rares (**solar4**). Dans un environnement où les contraintes sont faciles à satisfaire et où effectuer plusieurs sous-évaluations est nuisible (**solar7**), la méthode « s'éteint », elle trouve des assignations optimales qui sont équivalentes à ne pas utiliser la méthode et à ignorer les fidélités inférieures à 1. Cela implique que la méthode pourrait être ajoutée à un solveur existant où au pire, elle ne change rien et au mieux, elle améliore les solutions trouvées dans un temps donné. Il est important de préciser que cette dernière affirmation repose sur l'hypothèse que l'analyse du comportement des contraintes est de grande qualité, ce qui est difficile à garantir.

Un point important à rappeler est que la méthode ne gère pas la variabilité de la fonction objectif avec la fidélité autrement qu'en imposant une évaluation à $\phi = 1$ comme il est fait avec **solar7**. Il est possible d'ajouter une contrainte artificielle de la forme $f(x, \phi) \leq f^*$, et d'assigner cette contrainte selon une analyse du comportement de l'objectif avec la fidélité. Cela pose toutefois le risque de causer des interruptions très fréquentes, car durant une optimisation, il est rare de trouver un point meilleur que la meilleure solution depuis le début de l'optimisation. Une certaine proportion de points évalués sans interruption est nécessaire pour que la recherche directe trouve de bonnes directions. Cela dit, tel que montré au chapitre 2, plusieurs méthodes de gestion de la précision variable au niveau de la fonction objectif ont fait leurs preuves. Comme la méthode d'optimisation avec contraintes hiérarchisées ne modifie pas le solveur auquel elle est couplée, il est relativement simple de la combiner avec un algorithme qui lui gère la valeur de la fonction objectif variable. C'est d'ailleurs un avantage pour l'application de la méthode au problème de PRIAD, car pour réduire le temps des évaluations (d'environ 145 jours, voir section 1.1) à un temps raisonnable, plusieurs stratégies devront être combinées.

Pour terminer, la section 3.3 mentionne comme condition à l'application de la méthode la boîte noire possède des optimums locaux où des contraintes affectées par la multifidélité sont actives. Parmi les optimisations effectuées, lorsqu'il arrive qu'une implémentation de la méthode par contraintes hiérarchisée génère des résultats semblables au cas de base, c'est

souvent parce que les contraintes actives à l'optimum auquel converge la recherche directe ne sont pas des contraintes affectées par la multifidélité. Les interruptions sont alors rares. En particulier, ce phénomène survient souvent avec **solar3** et quelques fois avec **solar4**.

CHAPITRE 7 CONCLUSION ET RECOMMANDATIONS

7.1 Synthèse des travaux et limitations

Ce mémoire présente un algorithme d'optimisation applicable aux problèmes donnés par une boîte noire multifidélité sous contraintes. Cet algorithme doit être couplé avec un solveur existant, et il a pour but de réduire le temps des évaluations sur des points non réalisables en exploitant la multifidélité. Ainsi, de meilleures solutions sont obtenues pour un budget de temps donné. L'algorithme hiérarchise les contraintes en les assignant à différentes fidélités, de manière à estimer si un point évalué est non réalisable avec des lancements peu coûteux de la boîte noire. L'efficacité de la méthode est corrélée aux facteurs suivants :

- la rareté des points réalisables ;
- la pertinence des valeurs données par les contraintes à basse fidélité ;
- l'existence d'optimums locaux où les contraintes actives sont celles dont les valeurs sont pertinentes à basse fidélité ;
- l'homogénéité locale ou (préféablement) globale, du comportement des contraintes en fonction de la fidélité des points de l'espace des paramètres.

Les résultats des tests numériques montrent que le couplage de la méthode d'optimisation avec contraintes hiérarchisées avec le solveur NOMAD résulte en des solutions significativement meilleures pour des boîtes noires qui comportent ces propriétés. Si les points réalisables sont communs ou si les valeurs des contraintes aux basses fidélités sont impertinentes, la méthode a peu (et parfois aucun) impact sur l'optimisation. L'impact de l'état actif ou inactif des contraintes aux optimums locaux n'a pas été suffisamment étudié dans ce projet de recherche pour tirer des conclusions.

L'homogénéité du comportement des contraintes est la propriété la plus importante, car elle impacte directement la qualité de l'analyse effectuée à la première étape de l'algorithme. Les résultats numériques ont démontré que les plus grandes limitations de la méthode proviennent de cette analyse. Dans les pires cas, une mauvaise analyse cause une optimisation qui ne trouve aucun point réalisable, et ce, même si le point de départ est réalisable. Lorsque le comportement des contraintes en fonction de la fidélité est globalement homogène, il est suffisant d'échantillonner sur l'espace des paramètres. Autrement, il est nécessaire d'échantillonner localement, autour d'un point réalisable. En revanche, les résultats montrent que même pour les boîtes noires qui ont entraîné les résultats les plus problématiques, les solutions trouvées en couplant l'algorithme de contraintes hiérarchisées avec NOMAD sont grandement meilleures que celles trouvées par NOMAD sans couplage lorsqu'un échantillonnage local est effectué. Cela dit, l'utilisateur ne connaît pas nécessairement à priori de point réalisable,

et si c'est le cas, il ne sait pas nécessairement à quel degré il devrait échantillonner proche du point. De plus, si les comportements ne sont pas homogènes localement, l'analyse sera vraisemblablement tout de même mauvaise.

Lors de l'application de la méthode, il est nécessaire d'implémenter une phase de réalisabilité ainsi qu'une phase d'optimalité, car les tests ont révélé que l'algorithme de sous-évaluations interrompues est nuisible durant la première phase. Aussi, différentes méthodes de gestion des contraintes au niveau du solveur peuvent impacter l'optimisation. Les barrières progressive et extrême (BP et BE) ont été testées. Ces tests n'ont pas permis de conclure sur la barrière qu'il est préférable d'utiliser durant la phase de réalisabilité. Ce n'est toutefois pas problématique, car sachant que l'algorithme de sous-évaluations interrompues n'est pas actif durant la phase réalisabilité, on peut se fier sur la littérature pour affirmer que la BP est généralement préférable. Durant la phase d'optimalité, la BE est celle qui est la plus efficace dans la majorité des cas.

7.2 Améliorations futures

Les plus grandes améliorations potentielles ont trait à l'analyse du comportement des contraintes en fonction de la fidélité. Une méthode plus sophistiquée qu'un simple hypercube latin peut possiblement être développée. Entre autres, il serait intéressant d'implémenter la surveillance de comportements particuliers. À titre d'exemple, il peut être pertinent d'identifier les cas où une contrainte renvoie toujours la même valeur pour certaines fidélités, et de conclure que ces fidélités sont nécessairement non représentatives. Cela aurait évité les problèmes rencontrés avec la seconde contrainte des instances `solar3` et `solar4`. Aussi, au niveau du modèle d'optimisation de la matrice de biadjacence, des méthodes d'optimisation stochastique peuvent être investiguées. Par ailleurs, il pourrait être intéressant d'utiliser des outils de la théorie de la classification pour prédire si un point appartient au domaine réalisable Ω ou non. Cette prédiction pourrait servir à décider si l'algorithme 3.2 est lancé ou si plutôt une simple évaluation avec $\phi = 1$ est lancée, et ce, pour chaque point à évaluer. Sinon, cette prédiction pourrait être utile pour déterminer la séquence de sous-évaluations de l'algorithme 3.2 elle-même.

D'autres tests numériques pourraient être effectués. Principalement, d'autres solveur et d'autres boîtes noires apporteront des résultats plus variés. Aussi, certains paramètres n'ont pas été testés individuellement. Entre autres, des tests numériques peuvent être effectués avec différentes valeurs de ρ , ou différentes valeurs de ε . Les valeurs de ρ utilisées à la section 6.2.3 ont été choisies à l'intuition, ce sont les seules valeurs inférieures à 1 qui ont été testées, et elles ont donné d'excellents résultats. Il serait intéressant d'observer ce que différentes valeurs peuvent apporter. Finalement, lorsque la multifidélité d'une boîte noire est le résultat

de processus stochastiques, il est pertinent d'étudier des résultats avec différentes graines aléatoires. Durant ce projet, seulement l'impact de la graine aléatoire du solveur a été testé.

Enfin, plusieurs améliorations seront possibles lors de l'application de la méthode au problème provenant de PRIAD. Particulièrement, les assignations des contraintes aux fidélités pourront être dynamiques. Comme l'information à toutes les fidélités est accessible pour toute évaluation non interrompue, il sera possible de varier la matrice de biadjacence durant une optimisation en fonction du comportement des contraintes local observé. Cela n'est pas possible sans l'hypothèse de l'accès à l'information à toutes les fidélités. Aussi, avec PRIAD les contraintes sont posées sur des valeurs statistiques dont les distributions sont connues. Il est donc possible de faire l'usage d'estimateurs statistiques pour mieux déterminer la matrice de biadjacence.

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”, *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1150–1170, 2008. DOI : 10.1137/070707518. En ligne : <https://dx.doi.org/10.1137/070707518>

——, “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, édés. 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/PMAFS47429.2020.9183620. En ligne : <https://dx.doi.org/10.1109/PMAFS47429.2020.9183620>

K. Dzahini, M. Kokkolaras, et S. Le Digabel, “Constrained stochastic blackbox optimization using a progressive barrier and probabilistic estimates”, *Mathematical Programming*, no. 198, pp. 675–732, 2023. DOI : 10.1007/s10107-022-01787-7. En ligne : <http://dx.doi.org/10.1007/s10107-022-01787-7>

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, éd., 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, édés., vol. 115. PMLR, 22–25 Jul 2020, pp. 788–798. En ligne : <https://proceedings.mlr.press/v115/wu20a.html>

ANNEXE A Graphes réduits de différentes stratégies de calcul de B

Lors de la validation du calcul de la matrice de biadjacence B optimale de la section 6.1, les méthodes de calcul alternatives ont trouvé les graphes réduits suivants.

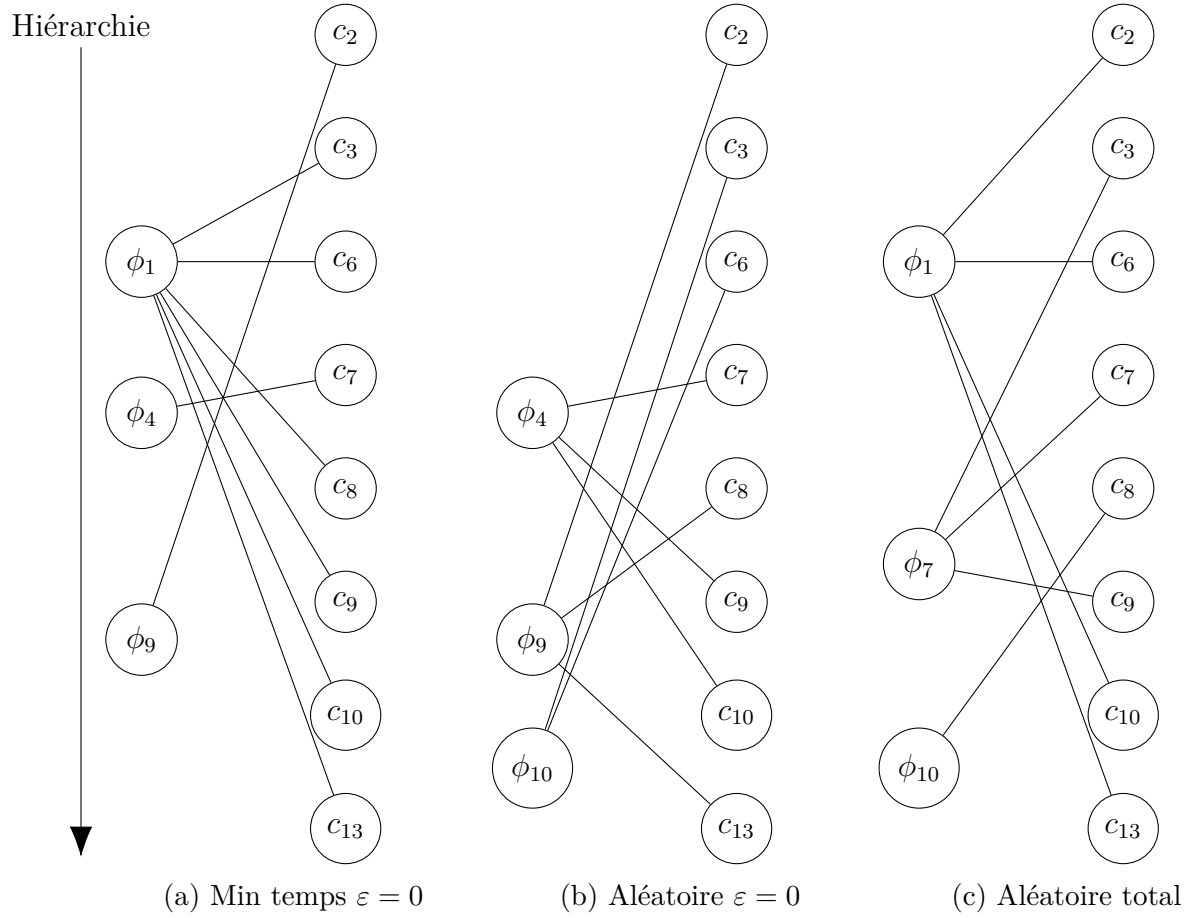


FIGURE A.1 Graphes réduits donnés par des stratégies alternatives pour **solar2**.

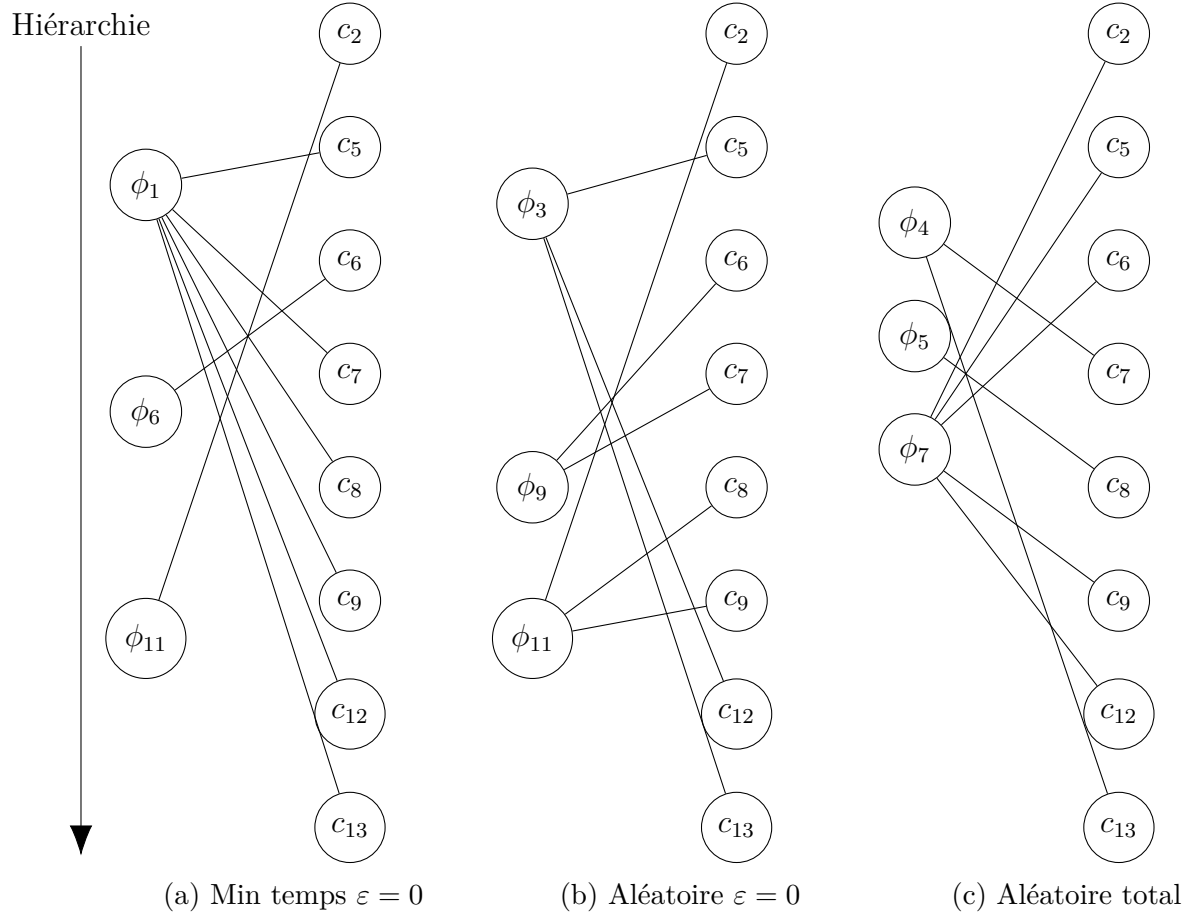


FIGURE A.2 Graphes réduits donnés par des stratégies alternatives pour **solar3**.

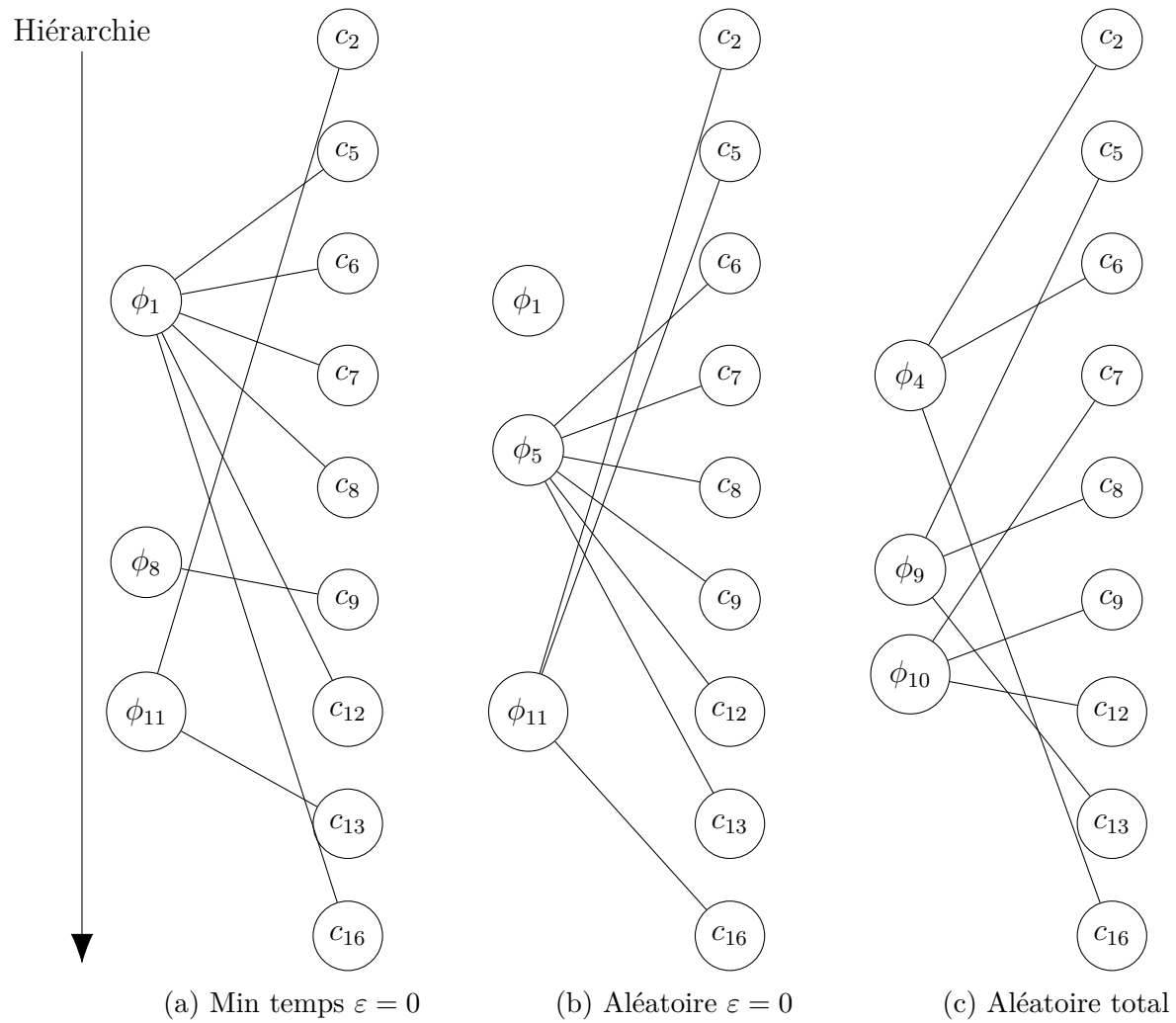


FIGURE A.3 Graphes réduits donnés par des stratégies alternatives pour **solar4**.

ANNEXE B Bornes des hypercubes latins

Les tableaux qui suivent montrent les bornes des différentes configurations d'hypercube latin présentées à la section 5.1. Des traits sont affichés lorsqu'une variable ne fait pas partie d'une instance.

TABLEAU B.1 Bornes des hypercubes latins avec $\rho = 1$.

	solar2		solar3		solar4		solar7	
Variable	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.
x_1	1	40	1	40	1	40	1	30
x_2	1	40	1	40	1	40	1	30
x_3	20	250	20	250	20	250	793	995
x_4	1	30	1	30	1	30	1	80
x_5	1	30	1	30	1	30	0.01	5
x_6	1	5300	1	500	1	2000	0.005	0.1
x_7	1	89	1	89	1	89	0.0055	0.1
x_8	0	20	0	20	0	20	-	-
x_9	1	20	1	20	1	20	-	-
x_{10}	793	995	793	995	793	995	-	-
x_{11}	1	72	1	50	1	50	-	-
x_{12}	0.01	5	1	30	1	30	-	-
x_{13}	0.005	0.1	0.01	5	0.01	5	-	-
x_{14}	0.005	0.1	0.01	5	0.01	5	-	-
x_{15}	-	-	495	650	495	650	-	-
x_{16}	-	-	1	80	1	1000	-	-
x_{17}	-	-	0.01	5	0.01	5	-	-
x_{18}	-	-	0.005	0.1	0.005	0.1	-	-
x_{19}	-	-	0.005	0.1	0.006	0.1	-	-

TABLEAU B.1 Bornes des hypercubes latins avec $\rho = 1$ (suite).

	solar2		solar3		solar4		solar7	
Variable	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.
x_{20}	-	-	1	8	0.007	0.2	-	-
x_{21}	-	-	-	-	0.5	10	-	-
x_{22}	-	-	-	-	0.005	0.1	-	-
x_{23}	-	-	-	-	0.006	0.1	-	-
x_{24}	-	-	-	-	0.15	0.4	-	-
x_{25}	-	-	-	-	2	6	-	-
x_{26}	-	-	-	-	1	24000	-	-
x_{27}	-	-	-	-	1	10	-	-
x_{28}	-	-	-	-	1	9	-	-
x_{29}	-	-	-	-	1	8	-	-

TABLEAU B.2 Bornes des hypercubes latins avec $\rho < 1$.

	solar2		solar3		solar4		solar7	
Variable	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.
x_1	1	19.75	14.1	21.9	20.05	23.95	1	8.27439
x_2	1.25	20.75	6.1	16.9	9.05	12.95	13.4927	27.9928
x_3	92.3	207.3	127	173	188.5	211.5	892.302	993.302
x_4	3.78	18.28	13.1	18.9	28.55	30	1	279
x_5	1	15.33	11.1	16.9	8.55	11.45	0.28387	2.77887
x_6	1676	4326	210	310	900	1100	0.04293	0.09043
x_7	164.89	89	55.2	72.8	74.6	83.4	0.06236	0.1
x_8	0	6.58	0	3.5	0	1.5	-	-
x_9	5.39	14.89	2.1	5.9	4.05	5.95	-	-
x_{10}	944.5	995	889.8	930.2	919.9	940.1	-	-
x_{11}	20	56	24.1	50	22.55	27.45	-	-
x_{12}	0.01	2.0275	6.1	11.9	28.55	30	-	-
x_{13}	0.005	0.04395	4.501	5	0.0505	0.5495	-	-
x_{14}	0.03435	0.08185	0.641	1.639	0.01	0.4495	-	-
x_{15}	-	-	634.5	650	552.25	564.75	-	-
x_{16}	-	-	47	63	580	680	-	-
x_{17}	-	-	3.501	4.5	1.0505	1.5495	-	-
x_{18}	-	-	0.005	0.0236	0.01175	0.02125	-	-
x_{19}	-	-	0.0905	0.1	0.0133	0.0227	-	-

TABLEAU B.2 Bornes des hypercubes latins avec $\rho < 1$ (suite).

	solar2		solar3		solar4		solar7	
Variable	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.	Inf.	Sup.
x_{20}	-	-	1	3	0.0073	0.02665	-	-
x_{21}	-	-	-	-	9.525	10	-	-
x_{22}	-	-	-	-	0.01075	0.02025	-	-
x_{23}	-	-	-	-	0.0113	0.0207	-	-
x_{24}	-	-	-	-	0.1875	0.2125	-	-
x_{25}	-	-	-	-	3	3	-	-
x_{26}	-	-	-	-	10800	13200	-	-
x_{27}	-	-	-	-	2	2	-	-
x_{28}	-	-	-	-	2	2	-	-
x_{29}	-	-	-	-	3	3	-	-