Résolution de problèmes d'optimisation pour les réseaux de transport d'électricité de grande taille avec des méthodes de programmation semi-définie positive

Julie Sliwak

UNIVERSITÉ SORBONNE PARIS NORD

Résolution de problèmes d'optimisation pour les réseaux de transport d'électricité de grande taille avec des méthodes de programmation semi-définie positive

THÈSE DE DOCTORAT EN COTUTELLE

présentée par

JULIE SLIWAK

Laboratoire d'Informatique de Paris Nord

Université Sorbonne Paris Nord

et

Département de mathématiques et de génie industriel

Polytechnique Montréal

pour obtenir le grade de

DOCTEUR EN INFORMATIQUE

soutenue le 22 juin 2021 devant le jury d'examen constitué de :

Dominique ORBAN	Polytechnique Montréal	Président du jury
Sourour ELLOUMI	ENSTA Paris	Rapportrice
Matthias TAKOUDA	Université Laurentienne	Rapporteur
Olivier BODINI	Université Sorbonne Paris Nord	Examinateur
Lucas LÉTOCART	Université Sorbonne Paris Nord	Directeur de thèse
Miguel F. ANJOS	Polytechnique Montréal	Co-directeur de thèse
Michel GENDREAU	Polytechnique Montréal	Co-directeur de thèse
Emiliano TRAVERSI	Université Sorbonne Paris Nord	Co-encadrant de thèse

REMERCIEMENTS

Je tiens tout d'abord à remercier mes directeurs de thèse Miguel F. Anjos, Lucas Létocart, Emiliano Traversi et Michel Gendreau qui m'ont apporté une aide précieuse tout au long de cette thèse. Je les remercie pour leur disponibilité, leurs conseils judicieux et leur soutien dans les moments difficiles.

Je tiens ensuite à remercier mes encadrants chez RTE Rémy Clément, Hélène Clémot, Tanguy Janssen et Anne-Claire Léger pour leur aide, leur bienveillance et pour tout ce qu'ils m'ont appris sur les réseaux de transport d'électricité. Un grand merci à Stéphane Fliscounakis qui m'a également aidée en me faisant profiter de son expertise en optimisation pour les réseaux de transport d'électricité.

Je remercie Françoise Sericola, responsable administrative de cette thèse chez RTE, sans qui la cotutelle entre l'université Sorbonne Paris Nord et Polytechnique Montréal n'aurait jamais pu se faire.

Je remercie Sourour Elloumi et Matthias Takouda d'avoir accepté d'être rapporteurs pour cette thèse. Je remercie Dominique Orban d'avoir accepté d'être le président de mon jury de thèse et Olivier Bodini d'avoir accepté d'être examinateur.

Je remercie les deux chefs du pôle Développement du Système, pôle au sein duquel je travaillais chez RTE, Gwilherm Poullennec puis Florent Xavier.

Un grand merci à Erling D. Andersen, Joachim Dahl et à toute l'équipe Mosek pour l'intérêt porté à mon travail, pour leur invitation dans leurs locaux à Copenhague et pour tous leurs conseils et bonnes idées.

Merci à Jean Maeght et Manuel Ruiz, chargés d'études R&D chez RTE, qui m'ont donné de bonnes idées au début de ma thèse.

Merci à Patrick Panciatici, conseiller scientifique chez RTE, et à Balthazar Donon, doctorant chez RTE, avec qui j'ai eu des discussions scientifiques passionnantes tout au long de ma thèse.

Merci à Pierre Carpentier, professeur à l'ENSTA Paris, qui m'a soutenue dans ce projet et qui a toujours été présent quand j'avais besoin de conseils.

Je tiens aussi à remercier mes collègues chez RTE pour ces trois années passées avec eux. Merci à Hadrien Godard et Clémentine Straub, anciens doctorants, pour leur bonne humeur au quotidien. Un grand merci à Gilles Bareilles, ancien stagiaire chez RTE, pour son aide sur

le module Julia développé au début de cette thèse. Merci à Marion Pichoud pour tous les bons moments passés ensemble. Merci à Fabiola Aravena, Marion Li, Mathieu Dussartre, Virginie Dussartre et Jean-Yves Bourmaud pour leur soutien sans faille. Merci à Mathieu Bague, Pauline Gambier-Morel, Claire Lajoie-Mazenc et Christine Yang pour toutes les discussions passionnantes que nous avons pu avoir.

Je remercie également mes collègues à l'université Sorbonne Paris Nord. Merci à Enrico Bettiol et Stefania Pan, anciens doctorants, qui ont su me faire profiter de leur expérience. Merci aux doctorants Étienne Leclercq et Juan Jose Torres Figueroa pour les bons moments passés ensemble, à l'université ou en conférence. Merci à mes collègues de bureau Sylvie Borne et Sophie Toulouse pour la bonne ambiance qui règne dans leur bureau et dont elles m'ont fait profiter. Merci à Charly Alizadeh, ingénieur stagiaire, qui explore avec enthousiasme une piste de recherche que je n'ai pas eu le temps de traiter pendant ma thèse.

Je tiens aussi à remercier les personnes que j'ai rencontrées durant mon année à Montréal et sans qui cette année n'aurait pas été aussi enrichissante. Merci à mes collègues de Polytechnique Montréal Alexis Montoison, Lisa Miron, Elizaveta Kuznetsova, Ludovic Salomon, Jeff Sylvestre-Décary, Florian Pedroli, Marie Pied, Pierre-Yves Bouchet, Vinicius Neves Motta, Tiphaine Bonniot, Marie-Ange Dahito, Alice Wu, Mathieu Besançon et Mathieu Tanneau qui m'ont aidée à apprécier mon bureau sans fenêtre. Merci à mon ancienne colocataire Karen Fernandes pour tous les bons moments passés ensemble. Merci à Christa et Jean-Pierre Frénois qui m'ont chaleureusement accueillie et qui m'ont aidée à m'acclimater.

Pour finir, je remercie mes proches qui m'ont apporté un soutien précieux tout au long de cette thèse.

RÉSUMÉ

Le problème de l'Optimisation des Flux de Puissance (OPF) consiste à déterminer la puissance à produire pour satisfaire la demande électrique à moindre coût, tout en tenant compte des contraintes du réseau de transport d'électricité. La résolution de ce problème est aujourd'hui encore un défi, principalement car le problème est non convexe. Récemment, des méthodes d'optimisation globale prometteuses ont été proposées pour résoudre le problème de l'OPF. La plupart de ces méthodes reposent sur la Programmation Semi-Définie (SDP) car les relaxations SDP de l'OPF sont très précises voire exactes. Cependant, la résolution de problèmes SDP de grande taille est très coûteuse, ce qui limite la portée de ces méthodes d'optimisation globale. L'objectif principal de ce projet de recherche est d'accélérer la résolution des relaxations SDP de l'OPF afin de montrer que l'optimisation SDP reste un outil puissant pour l'OPF et ce, même pour des réseaux de grande taille.

Pour atteindre cet objectif, nous commençons par concevoir des outils pour faciliter l'implémentation du problème de l'OPF et de ses nombreuses variantes. Nous cherchons ensuite à accélérer la résolution de la relaxation du rang de l'OPF en exploitant des méthodes de décomposition en cliques maximales. Nous proposons une nouvelle stratégie de fusion de cliques qui permet d'améliorer significativement le temps de résolution pour les instances MATPOWER. Nous proposons finalement une méthode d'énumération implicite s'appuyant sur une relaxation SDP pour résoudre différents problèmes d'OPF réactif (des problèmes d'OPF avec des variables binaires). Notre méthode permet de calculer la solution optimale ou au moins une bonne solution réalisable pour toutes les instances MATPOWER, y compris pour les plus gros réseaux. De plus, elle calcule de meilleures solutions qu'une heuristique d'arrondi, dans une limite de temps raisonnable et de manière robuste.

Ainsi, nous montrons dans cette thèse qu'il est possible d'accélérer significativement la résolution de la relaxation du rang de l'OPF, la principale relaxation SDP utilisée dans le contexte de l'OPF. Nous démontrons aussi que l'optimisation SDP est un outil puissant pour résoudre des problèmes d'OPF réactif et cela même pour des réseaux de plus de 6000 nœuds.

ABSTRACT

The Optimal Power Flow (OPF) is a famous power system optimization problem. It consists in computing an optimal power generation dispatch for an alternating current transmission network that respects power flow equations and operational constraints. The OPF problem is highly nonconvex and there is still no efficient method to solve this problem to global optimality. Nevertheless, promising methods have been proposed recently. Most of them are based on Semidefinite Programming (SDP) since semidefinite relaxations of OPF problems are tight. However, the resolution of semidefinite relaxations is very costly, especially for large-scale networks. Therefore, all promising methods depend on large-scale SDP problems being solved efficiently. The main objective of this research project is to decrease the resolution time of semidefinite relaxations of OPF problems to demonstrate that semidefinite relaxations are still powerful to solve OPF problems, even for large-scale networks.

To reach this objective, we first conceive programming tools to facilitate the implementation of the OPF problem and its variants. We then seek to speed up the resolution of the rank relaxation for OPF problems thanks to clique decomposition techniques. We propose a new strategy of clique merging which significantly improves the solving time for MATPOWER instances. We finally propose a SDP-based Branch-and-Bound algorithm to solve several Reactive OPF problems that contain binary variables. Our algorithm computes the optimal solution or at least a good feasible solution for all MATPOWER instances, including for large-scale networks. Moreover, it computes better solutions than a rounding heuristic and it is also more robust.

Thus, we show in this thesis that it is possible to significantly speed up the resolution of the rank relaxation of the OPF problem, which is the main SDP relaxation used to solve OPF problems. We also demonstrate that semidefinite programming is a powerful tool to solve Reactive Optimal Power Flow problems, even for networks with more than 6000 buses.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ADMM	Méthode des Multiplicateurs à Direction Alternée
AMD	Approximation du Degré Minimal
GO	Grid Optimization
IP	Point intérieur
KKT	Karush-Kuhn-Tucker
MCS	Moyen de Compensation Statique
MD	Degré Minimal
MINLP	Problème Non Linéaire en variables MIxtes
OPF	Optimisation des Flux de Puissance
POP	Problème d'Optimisation Polynomial
PSCOPF	Optimisation des Flux de Puissance Préventif avec Contraintes de Sécurité
QCQP	Problème Quadratique avec Contraintes Quadratiques
ROPF	Optimisation des Flux de Puissance Réactif
SDP	Programmation Semi-définie positive
SOCP	Programmation sur les Cônes de Second Ordre

CHAPITRE 1 INTRODUCTION

Le problème de l'Optimisation des Flux de Puissance (OPF) en Courant Alternatif (AC) est un problème d'optimisation dans les réseaux de transport d'électricité étudié depuis 1962 [2]. Il consiste à déterminer la puissance à produire pour satisfaire la demande en électricité en minimisant les coûts de production tout en respectant des contraintes réseaux telles que des contraintes de transit de puissance sur les ouvrages ou de qualité de la tension. Ce problème est NP-difficile [3,4], ce qui signifie qu'il est au moins aussi difficile que les problèmes les plus difficiles de la classe NP. La classe NP (Polynomiale Non déterministe) est une classe très importante de la théorie de la complexité qui rassemble les problèmes de décision pour lesquels il est possible de vérifier en temps polynomial si une solution candidate est effectivement solution.

La résolution exacte de l'OPF en AC est un enjeu majeur pour RTE (Réseau de Transport d'Électricité), le gestionnaire du réseau de transport d'électricité français, partenaire industriel de ce projet de recherche. En effet, dans le découpage en Europe entre les activités libéralisées de production/fourniture et des activités régulées de gestion du réseau, c'est le rôle des gestionnaires comme RTE d'anticiper les contraintes réseaux et aider à la gestion de ces contraintes aux différents horizons depuis le court terme dans le cadre de l'exploitation (e.g. analyses de sécurités) au long terme dans le cadre des études d'évolution du réseau amenant aux décisions d'investissement et de gestion des actifs. La production et demande d'électricité sont quant à elles pilotées dans le cadre des marchés via des optimisations qui peuvent alors s'appuyer sur une modélisation très simplifiée du réseau, par exemple sous forme de capacités d'échanges Flow-Based entre zones de prix pour les marchés journaliers.

Le réseau de transport d'électricité permet de relier les producteurs aux consommateurs d'électricité. En France et en Europe, la production d'électricité est pilotée par un problème d'optimisation plus simple que le problème de l'OPF puisqu'il ne modélise pas le réseau de transport d'électricité. Cependant, les problèmes d'OPF sont souvent utilisés à RTE comme sous-problèmes dans des chaînes de simulation du système électrique, en particulier dans des études long terme pour le renforcement de réseau. Des applications court terme sont aussi étudiées : des problèmes d'OPF pourraient être utilisés pour simuler le comportement des dispatchers, les aiguilleurs du réseau électrique qui pilotent la circulation des flux électriques.

La problématique étudiée dans ce projet de recherche est détaillée ci-dessous, après une brève présentation des concepts de base qui seront utilisés tout au long de cette thèse.

1.1 Définitions et concepts de base

1.1.1 Optimisation

Un problème d'optimisation est un problème de la forme :

$$\begin{array}{ll} \min & f(x) \\ s.c. & g(x) \leq 0 \\ & x \in \Omega \end{array} \tag{1.1}$$

avec Ω un ensemble fermé de $\mathbb{R}^n$, f une fonction de Ω dans $\mathbb{R}$ et g une fonction de Ω dans $\mathbb{R}^m$. Ce problème consiste à trouver un élément $x^* \in \Omega$ qui minimise la fonction f tout en respectant les contraintes $g(x^*) \leq 0$. La valeur $f(x^*)$ est alors la valeur optimale du problème. La fonction f est appelée la fonction objectif. L'ensemble $R = \{x \in \Omega : g(x) \leq 0\}$ est fermé et est appelé ensemble des solutions réalisables. Si l'ensemble R est vide, le problème est dit non réalisable.

Un élément $x \in R$ est un *minimum global* si $\forall y \in R, f(x) \leq f(y)$. Un élément $x \in R$ est un *minimum local* s'il existe $\varepsilon > 0$ tel que $\forall y \in R$ avec $||y - x|| < \varepsilon$, $f(x) \leq f(y)$. Dans le cas convexe, c'est-à-dire que la fonction f est convexe et l'ensemble R est convexe, tout minimum local est un minimum global. Dans le cas non convexe (f non convexe et/ou R non convexe), rien ne garantit qu'un minimum local soit global. Un minimum local permet seulement de majorer la valeur optimale du problème. C'est pourquoi résoudre un problème d'optimisation non convexe peut s'avérer très compliqué : il existe plusieurs méthodes pour calculer un minimum local mais il faut disposer de méthodes exactes plus coûteuses pour calculer le minimum global.

1.1.2 Relaxation d'un problème d'optimisation

L'intérêt de relâcher un problème d'optimisation est d'obtenir un problème plus simple à résoudre. Pour cela, il est possible d'étendre l'ensemble réalisable et/ou de sous-estimer la fonction objectif. Une relaxation du problème d'optimisation 1.1 est un problème de la forme :

$$\begin{array}{ll} \min & \underline{f}(x) \\ s.c. & \underline{g}(x) \leq 0 \\ & x \in \underline{\Omega} \end{array} \tag{1.2}$$

avec $\underline{\Omega}$ un ensemble fermé de $\mathbb{R}^n$, $\underline{f}$ une fonction de $\underline{\Omega}$ dans $\mathbb{R}$ et $\underline{g}$ une fonction de $\underline{\Omega}$ dans $\mathbb{R}^m$ tels que $R \subseteq \underline{R} = \{x \in \underline{\Omega} : \underline{g}(x) \leq 0\}$ et $\underline{f}(x) \leq f(x)\ \forall x \in R$. La valeur optimale d'une

relaxation fournit une borne inférieure de la valeur optimale du problème initial.

Ce concept est très utile dans le cadre de l'optimisation non convexe. Il est en effet judicieux de relâcher un problème non convexe en un problème convexe (i.e. $\underline{f}$ et $\underline{R}$ convexes) pour obtenir des informations sur la valeur optimale du problème non convexe. Les relaxations convexes sont donc très utilisées dans les méthodes exactes qui cherchent à calculer le minimum global d'un problème non convexe.

1.2 Éléments de la problématique

Malgré les nombreux travaux de recherche sur le sujet, la résolution du problème de l'OPF en AC pour les réseaux de transport d'électricité reste un défi, principalement parce que le problème est non convexe. La situation est un peu différente pour les réseaux de distribution car ces réseaux ont généralement une structure arborescente, ce qui facilite la résolution [5]. Ce n'est pas le cas des réseaux de transport d'électricité ; ceux-ci contiennent en effet des cycles pour respecter la règle du N-1 qui stipule que le réseau doit rester fonctionnel malgré la perte d'une ligne. Il existe cependant de nombreuses méthodes locales pour l'optimisation non linéaire qui permettent d'obtenir de bonnes solutions réalisables. Autrement dit, il est relativement facile de calculer des bons majorants de la valeur optimale du problème de l'OPF. RTE a déjà appuyé deux doctorats pour avancer sur la résolution globale de l'OPF en AC. Le premier s'est intéressé à la hiérarchie de Lasserre qui consiste à résoudre une suite de relaxations semi-définies positives [6,7]. Le deuxième doctorat a exploré la piste d'une méthode d'énumération implicite [8, 9] à partir d'une reformulation quadratique s'appuyant sur une relaxation semi-définie positive. Ces deux méthodes, comme d'autres méthodes d'optimisation globale [10], reposent sur la Programmation Semi-Définie (SDP). L'utilisation courante des relaxations SDP est spécifique au domaine de l'OPF et s'explique par le fait que les relaxations SDP fournissent des minorants de très bonne qualité pour l'OPF en AC. Elles sont donc aussi utilisées si l'on ne cherche qu'à obtenir un bon encadrement de la valeur optimale du problème. Toutefois, la résolution des problèmes SDP n'est efficace que pour les problèmes de taille moyenne. Ainsi, de nombreuses méthodes de résolution globale sont limitées par la taille des problèmes SDP.

Ce projet de recherche s'inscrit dans le contexte de l'optimisation semi-définie positive à grande échelle [11,12]. L'optimisation semi-définie positive est une généralisation de l'optimisation linéaire au cas matriciel : un problème SDP est un problème de minimisation convexe dans lequel la variable est une matrice à valeurs réelles. Celle-ci doit appartenir au cône des matrices semi-définies positives (matrices symétriques à valeurs propres positives ou nulles) et elle doit aussi satisfaire des contraintes affines. La fonction objectif d'un problème SDP est

une fonction linéaire. Les problèmes SDP peuvent aussi être définis en nombres complexes avec des matrices hermitiennes. L'optimisation SDP peut être utilisée dans le contexte de l'optimisation non convexe pour obtenir des relaxations convexes de bonne qualité.

Les problèmes SDP d'une taille raisonnable (quelques milliers de variables et quelques dizaines de milliers de contraintes si le problème est creux) peuvent être résolus de manière efficace en temps polynomial avec des algorithmes de points intérieurs [13]. Toutefois, les problèmes SDP de grande taille ne sont pas résolus par les meilleurs algorithmes de points intérieurs, principalement à cause de la matrice hessienne du Lagrangien qu'il faut construire, stocker en mémoire et factoriser. Ceci est vrai même pour les problèmes SDP creux car la matrice hessienne ne conserve pas le caractère creux d'un problème. Autrement dit, la matrice hessienne d'un problème SDP creux peut être dense. Pour pallier ce problème, d'autres méthodes de résolution sont étudiées dans le cas où le problème est creux. Il est possible de définir trois grandes familles pour ces méthodes. Les méthodes de la première famille se concentrent sur les améliorations des algorithmes de points intérieurs, e.g. en cherchant des matrices de préconditionnemment efficaces [14]. Les méthodes de la deuxième famille consistent à reformuler le problème SDP de grande taille avec des matrices semi-définies positives de plus petite taille, les matrices SDP de petite taille étant mieux gérées dans un algorithme de points intérieurs. Ces méthodes reposent sur des algorithmes d'extension cordale et de recherche de cliques maximales [15,16]. Ces méthodes peuvent aussi être combinées à des méthodes de réduction faciale [17]. La troisième famille rassemble les méthodes de premier ordre qui sont moins coûteuses que les méthodes de points intérieurs car elles n'utilisent que les dérivées de premier ordre. En contrepartie, elles peuvent être moins précises et/ou connaître des problèmes de convergence. Les méthodes de premier ordre les plus connues sont les méthodes ADMM (Méthode des Multiplicateurs à Direction Alternée) [18] ou les méthodes de type Dantzig-Wolfe [19]. D'autres méthodes de premier ordre ont été proposées pour résoudre les relaxations SDP de l'OPF [20]. Certaines de ces méthodes sont définies sur le problème SDP reformulé avec des matrices SDP de petite taille, ce qui permet d'accélérer la résolution puisque l'on travaille alors sur des petits sous-problèmes indépendants (et donc parallélisables). Toutes ces méthodes font l'objet de nombreuses recherches à ce jour.

1.3 Objectifs de recherche

L'objectif principal de ce projet de recherche est d'accélérer la résolution des relaxations SDP de l'OPF afin de montrer que l'optimisation SDP reste un outil puissant pour l'OPF même pour des réseaux de grande taille. Nous pouvons découper cet objectif en trois sous-objectifs :

— Concevoir et développer des outils génériques afin de faciliter l'implémentation du

problème de l'OPF et de ses variantes ;
— Accélérer la résolution de la relaxation du rang pour l'OPF sans faire de compromis sur la précision ;
— Montrer que l'accélération de la résolution de la relaxation du rang permet d'utiliser une relaxation SDP à l'intérieur d'une méthode d'énumération implicite pour résoudre des variantes du problème de l'OPF avec des variables binaires.

1.4 Plan de la thèse

Cette thèse est organisée de la façon suivante. La revue de littérature est présentée dans le chapitre 2. Elle débute par une liste de notations suivie d'un état de l'art sur le problème de l'OPF puis sur l'optimisation semi-définie positive. Une brève présentation du langage Julia termine ce chapitre. Le chapitre 3 est consacré aux outils que nous avons conçus et développés pour répondre aux besoins de cette thèse. Nous présentons d'abord un module Julia pour l'optimisation polynomiale en variables complexes puis un module générique pour le problème de l'OPF. Nous testons ces modules sur une variante du problème de l'OPF qui prend en compte des contraintes de sécurité. Le chapitre 4 expose les travaux relatifs à l'accélération de la résolution des relaxations SDP de l'OPF. Ce chapitre commence par un état de l'art des outils disponibles pour résoudre la relaxation du rang de l'OPF. Nous montrons ensuite dans ce chapitre que le choix de la décomposition en cliques maximales peut avoir un impact significatif sur le temps de résolution de la relaxation du rang et nous proposons une nouvelle stratégie de résolution. Nous finissons par une extension aux relaxations SDP issues de la hiérarchie de Lasserre d'ordre 2. Le chapitre 5 démontre l'intérêt de l'optimisation semi-définie positive pour un problème d'OPF étendu, le problème de l'OPF réactif. Nous présentons d'abord les trois variantes du problème de l'OPF réactif qui nous intéressent. Nous détaillons ensuite la relaxation SDP utilisée, la méthode de résolution qui s'appuie dessus et les résultats obtenus. Pour finir, le chapitre 6 présente une synthèse des résultats obtenus et les perspectives d'amélioration.

CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre présente un état de l'art sur le problème de l'OPF et sur l'optimisation semi-définie positive. Il commence par une description des notations utilisées tout au long de cette thèse et se termine sur une brève présentation du langage Julia.

2.1 Notations

Cette section présente toutes les notations mathématiques utilisées dans la suite de ce document :

- $\mathbb{R}$ désigne l'ensemble des nombres réels ;
- $\mathbb{R}^n$ désigne l'ensemble des vecteurs de taille n à composantes dans $\mathbb{R}$;
- $||v||_2$ désigne la norme euclidienne du vecteur $v \in \mathbb{R}^n$;
- $\mathbb{C}$ désigne l'ensemble des nombres complexes ;
- $\mathbb{C}^n$ désigne l'ensemble des vecteurs de taille n à composantes dans $\mathbb{C}$;
- $\mathbf{j}$ désigne le nombre imaginaire tel que $\mathbf{j}^2 = -1$;
- $\overline{c}$ désigne le conjugué du nombre complexe $c \in \mathbb{C}$;
- $\overline{v}$ désigne le vecteur conjugué du vecteur complexe $v \in \mathbb{C}^n$;
- $Re(z)$ désigne la partie réelle du nombre complexe z ;
- $Im(z)$ désigne la partie imaginaire du nombre complexe z ;
- la transposée d'un vecteur complexe ou réel v ou d'une matrice complexe ou réelle M est notée v^T ou M^T ;
- la transposée conjuguée d'un vecteur complexe v ou d'une matrice complexe M est notée v^H ou M^H ;
- $\mathbb{S}^n$ désigne l'ensemble des matrices symétriques réelles de taille $n \times n$;
- la notation $X \succeq 0$ est utilisée pour signifier qu'une matrice X, réelle ou complexe, est semi-définie positive ;
- $\mathbb{S}^n_+$ désigne l'ensemble des matrices semi-définies positives réelles de taille $n \times n$;
- $A \cdot B$ désigne le produit scalaire matriciel classique pour les matrices A et B, i.e., $A \cdot B = trace(A^T B)$.

2.2 Le problème de l'Optimisation des Flux de Puissance

Il existe plusieurs formulations du problème de l'OPF. Dans la suite de cette section, nous présentons la version qui est la plus intéressante du point de vue de RTE.

2.2.1 Modélisation

Modélisation d'un réseau de transport d'électricité

Un réseau de transport d'électricité permet d'acheminer l'électricité produite par des producteurs d'électricité (centrales nucléaires, panneaux solaires, éoliennes, etc) vers des consommateurs (distributeurs d'électricité ou consommateurs industriels). D'un point de vue mathématique, il peut être représenté par un graphe orienté dans lequel les sommets représentent des nœuds électriques qui peuvent consommer et/ou produire de l'électricité. Les arcs représentent des lignes de transmission et/ou des transformateurs. La figure 2.1 présente un exemple de réseau avec 9 nœuds dont 3 nœuds producteurs et 3 nœuds consommateurs. La plupart des lignes de transmission sont en Courant Alternatif (AC) et elles fonctionnent à haute tension pour réduire les pertes par effet Joule. D'un point de vue mathématique, un courant alternatif est généralement décrit par une fonction sinusoïdale qui dépend du temps. Si l'on se place à un instant donné, le courant alternatif peut être décrit par un nombre complexe. Les autres grandeurs physiques comme la tension et la puissance peuvent aussi être décrites par des nombres complexes. Ainsi, dans toute la suite de cette section, nous définissons le problème en nombres complexes. Par ailleurs, à cause des pertes par effet Joule, l'intensité et la puissance définies sur une ligne de transmission varient entre le nœud origine et le nœud destination. C'est pourquoi nous définissons deux quantités pour chacune de ces grandeurs : une intensité et une puissance à l'origine de chaque ligne et une intensité et une puissance à la destination de chaque ligne.

Soit $T = (N, B)$ un réseau de transport d'électricité et $G \subset N$ l'ensemble des nœuds producteurs. Résoudre un problème d'OPF consiste à déterminer, pour un instant donné, les valeurs des grandeurs physiques suivantes :

- la tension complexe v_n en chaque nœud $n \in N$;
- la puissance de production complexe en chaque nœud $n \in N$, notée S_n^{gen}. Celle-ci vaut 0 pour les nœuds qui ne sont pas producteurs, i.e., $S_n^{gen} = 0 \ \forall n \notin G$;
- l'intensité et la puissance complexes à l'origine de chaque ligne $l \in B$, notées respectivement i_l^{orig} et S_l^{orig} ;
- l'intensité et la puissance complexe à l'extrémité de chaque ligne $l \in B$, notées respectivement i_l^{dest} et S_l^{dest}.

Il existe des liens entre ces grandeurs physiques.

Une puissance complexe S s'écrit en représentation cartésienne $S = P + \mathbf{j}Q$ avec P la partie réelle de la puissance que l'on appelle la puissance active et Q la partie imaginaire que l'on appelle la puissance réactive.

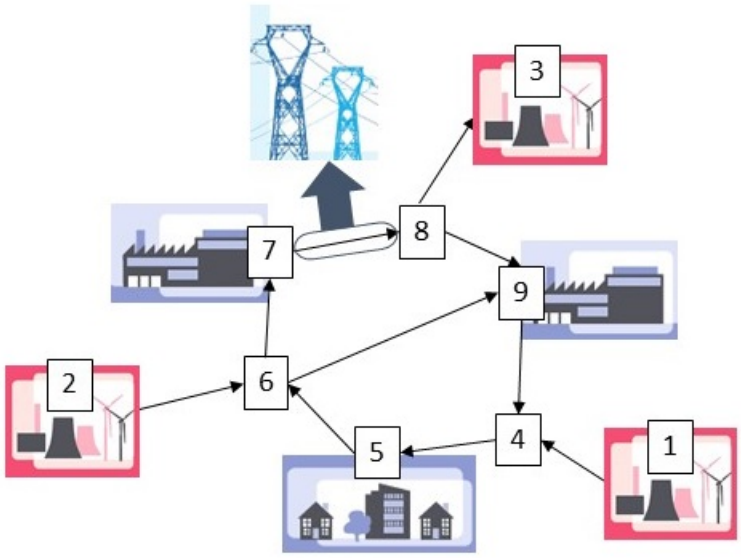

Figure 2.1 Réseau de transport d'électricité avec 9 nœuds dont 3 producteurs. Image créée à partir de l'image disponible sur le site **RTE & vous** à la page Notre rôle au quotidien Transporter, équilibrer et optimiser (page consultée le 19/07/2019 à 14h36).

Modélisation d'une ligne de transmission Une ligne électrique est modélisée par un « quadripôle en π » précédé d'un transformateur (action sur le module et/ou la phase de la tension). La figure 2.2 détaille cette modélisation pour une ligne électrique l allant du nœud o au nœud d. Voir la documentation MATPOWER [21] pour plus de détails.

Les paramètres physiques associés à une ligne électrique l sont :

- une résistance r_l ;
- une réactance x_l ;
- une susceptance de charge b_l ;
- un ratio de transformateur τ_l ;
- un angle de déphasage de transformateur θ_l.

La résistance r_l et la réactance x_l définissent ce que l'on appelle l'impédance complexe z_l du modèle en π : $z_l = r_l + \mathbf{j}x_l$. L'admittance en série y_l est l'inverse de l'impédance complexe : $y_l = \frac{1}{z_l}$. La matrice d'admittance Y_l prend en compte les paramètres physiques de la modélisation π (y_l, b_l) et ceux du transformateur (τ_l, θ_l). Elle est définie de la façon suivante :

$$Y_l = \begin{pmatrix} \frac{y_l + \mathbf{j}b_l}{\tau_l^2} & -y_l \frac{1}{\tau_l e^{-\mathbf{j}\theta_l}} \\ -y_l \frac{1}{\tau_l e^{\mathbf{j}\theta_l}} & y_l + \mathbf{j}b_l \end{pmatrix}. \tag{2.1}$$

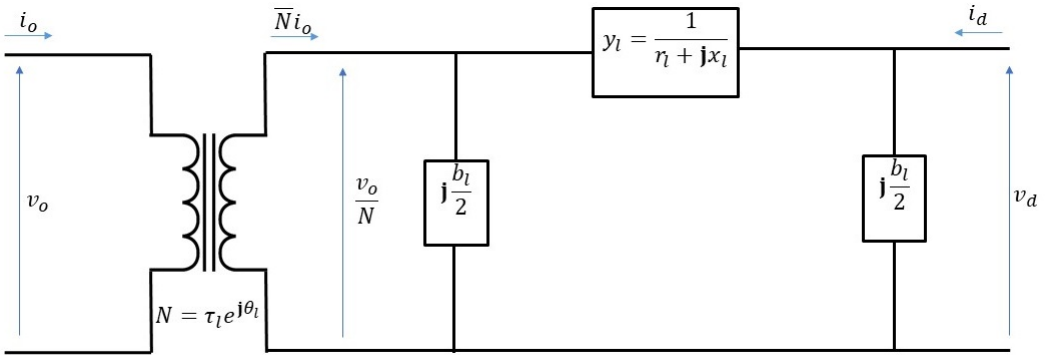

Figure 2.2 Schéma d'une ligne de transport en π - Image du transformateur créée par jjbeard et disponible publiquement ici

Pour chaque ligne $l = (o, d)$, la matrice d'admittance Y_l permet de définir l'intensité en fonction de la tension de la façon suivante :

$$\begin{pmatrix} i_l^{orig} \\ i_l^{dest} \end{pmatrix} = Y_l \begin{pmatrix} v_o \\ v_d \end{pmatrix}. \tag{2.2}$$

Définition de la puissance La puissance complexe à l'origine ou à la destination d'une ligne est définie à partir de la tension et de l'intensité. Plus précisément, sur une ligne $l = (o, d)$ qui va du nœud o au nœud d, on a :

$$S_l^{orig} = v_o \overline{i_l^{orig}}, \tag{2.3}$$

$$S_l^{dest} = v_d \overline{i_l^{dest}}. \tag{2.4}$$

Il est possible d'exprimer la puissance sur les lignes uniquement en fonction de la tension grâce aux relations (2.2), (2.3) et (2.4) :

$$S_l^{orig} = v_o(\overline{(Y_l)_{11} v_o + (Y_l)_{12} v_d}) = \frac{\overline{y_l} - \mathbf{j}b_l}{\tau_l^2}|v_o|^2 - \overline{y_l}\frac{1}{\tau_l e^{\mathbf{j}\theta_l}} v_o \overline{v_d}, \tag{2.5}$$

$$S_l^{dest} = v_d(\overline{(Y_l)_{21} v_o + (Y_l)_{22} v_d}) = -\overline{y_l}\frac{1}{\tau_l e^{-\mathbf{j}\theta_l}} \overline{v_o} v_d + (\overline{y_l} - \mathbf{j}b_l)|v_d|^2. \tag{2.6}$$

Ainsi, les tensions permettent d'exprimer les puissances et les intensités sur les lignes. En pra-

tique, il suffit donc de déterminer les tensions et les puissances de production pour connaître toutes les grandeurs physiques sur le réseau.

Moyens de Compensation Statique D'autres éléments peuvent être associés aux nœuds électriques : des Moyens de Compensation Statique (MCS) qui permettent de contrôler la puissance réactive. Il est important de pouvoir contrôler cette puissance afin d'éviter des problèmes de tension. Dans la version classique de l'OPF, les MCS sont des objets passifs que l'optimiseur ne cherche pas à contrôler. Dans d'autres variantes de l'OPF, les MCS peuvent être mis en service ou hors service à l'aide de variables binaires, ce qui permet de contrôler la puissance réactive. Les MCS peuvent consommer ou produire de la puissance à un nœud n selon deux paramètres : une conductance g_n et une susceptance b_n. En général, la conductance g_n est nulle et le MCS n'agit que sur la puissance réactive en fonction de la tension au nœud. La puissance induite par un MCS est la suivante :

$$S_n^{MCS} = (g_n - \mathbf{j}b_n)|v_n|^2. \tag{2.7}$$

Modèle d'optimisation

Données : Les données d'un problème d'OPF sont :

- un ensemble de nœuds N, un ensemble de nœuds producteurs $G \subset N$, un ensemble de MCS $S \subset N$ et un ensemble de lignes de transmission B ;
- des paramètres physiques sur chaque ligne $l \in B$ (résistance, réactance, susceptance de charge, ratio de transformateur, angle de déphasage de transformateur) ;
- des paramètres de conductance g_n et susceptance b_n pour chaque MCS $n \in S$;
- la puissance électrique complexe demandée en chaque nœud $n \in N$, notée S_n^d ;
- des bornes sur le module de tension pour chaque nœud $n \in N$, notées v_n^{min} et v_n^{max} ;
- des bornes sur la puissance active produite par chaque nœud producteur $g \in G$, notées P_g^{min} et P_g^{max} ;
- des bornes sur la puissance réactive produite par chaque nœud producteur $g \in G$, notées Q_g^{min} et Q_g^{max} ;
- des limites thermiques sur chaque ligne $l \in B$, notées i_l^{max} pour une modélisation en intensité ;
- une fonction coût c_g (linéaire ou quadratique) pour chaque nœud producteur $g \in G$.

Variables : Les variables d'un problème d'OPF sont :

- la tension complexe v_n en chaque nœud n ;

— la puissance complexe S_g^{gen} fournie par chaque nœud producteur $g \in G$.

Contraintes : Il existe plusieurs types de contraintes dans un problème d'OPF.

— Bilans de puissance :
Les contraintes de bilans de puissance sont les contraintes issues de la physique. Elles sont quadratiques en la tension. Plus précisément, elles n'impliquent que des produits hermitiens. En chaque nœud, la contrainte de bilan de puissance exige que la puissance qui entre dans le nœud (puissance générée) soit égale à la puissance qui sort du nœud (puissance demandée, puissance des lignes, puissance liée aux MCS). Plus précisément elles s'écrivent pour un nœud n :

$$S_n^{gen} = S_n^d + \sum_{l=(o,n)} S_l^{dest} + \sum_{l=(n,d)} S_l^{orig} + (g_n - jb_n)|v_n|^2 \tag{2.8}$$

où S_l^{dest} et S_l^{orig} dépendent uniquement de la tension d'après (2.6) et (2.5). Rappel : si le nœud n considéré n'est pas producteur alors $S_n^{gen} = 0$.
Ces contraintes sont principalement responsables de la non-convexité du problème de l'OPF.

— Contraintes de sécurité sur la tension :
Pour éviter de détériorer le matériel (tension trop haute) et l'écroulement de tension (tension trop basse), les gestionnaires de transport d'électricité définissent des plages normales de fonctionnement pour le module de la tension. Le module de la tension est ainsi borné en chaque nœud n :

$$v_n^{min} \leq |v_n| \leq v_n^{max}. \tag{2.9}$$

Si la représentation cartésienne des variables complexes est utilisée, ces contraintes introduisent aussi de la non-convexité dans le problème (contraintes de la forme $x^2 + y^2 \geq a^2$).

— Contraintes de production :
Tous les groupes de production ont des limites physiques en puissance qu'ils ne peuvent dépasser. Ainsi, pour chaque nœud producteur $g \in G$, la puissance active et la puissance réactive sont bornées :

$$P_g^{min} \leq Re(S_g^{gen}) \leq P_g^{max}, \tag{2.10}$$

$$Q_g^{min} \leq Im(S_g^{gen}) \leq Q_g^{max}. \tag{2.11}$$

Il faut noter que nous utilisons un diagramme de groupe PQ simplifié dans cette modélisation : nous utilisons un diagramme rectangulaire et non un parallélogramme avec les bornes sur la puissance réactive qui dépendent de la puissance active.

— Limites thermiques sur les lignes :
Les limites thermiques modélisent l'échauffement des lignes. Elles peuvent être exprimées comme des contraintes de capacité maximale en puissance apparente ou en intensité. Dans notre cas, nous modélisons les limites thermiques en intensité car c'est ce qui modélise le mieux les contraintes réelles de RTE.
Pour chaque ligne l, les limites thermiques concernent le module de l'intensité :

$$\begin{cases} |i_l^{orig}| \leq i_{,}^{max} \\ |i_l^{dest}| \leq i_l^{max} \end{cases} \tag{2.12}$$

où i_l^{orig} et i_l^{dest} dépendent uniquement de la tension d'après (2.2).

Objectif : Un problème d'OPF a généralement pour objectif de minimiser les coûts de production ou de minimiser les pertes. Dans les deux cas, ceci revient à une somme de fonctions quadratiques ou linéaires c_g en la puissance active pour chaque nœud producteur $g \in G$. L'objectif est donc la fonction réelle suivante :

$$\sum_{n \in G} c_g(Re(S_n^{gen})). \tag{2.13}$$

Si les fonctions sont quadratiques en la puissance active, elles sont quartiques en la tension. Si les fonctions sont linéaires en la puissance active, l'objectif est quadratique en la tension. Chez RTE, nous utilisons des fonctions linéaires car l'objectif est de minimiser la déviation par rapport à un plan de production défini auparavant.

Modèle : Pour résumer, un problème d'OPF peut être écrit sous forme de Problème Quadratique avec Contraintes Quadratiques (QCQP) en variables complexes :

$$
\begin{aligned}
\min \quad & \sum_{g\in G} c_g(Re(S_g^{gen})) & \\
s.c. \quad & S_n^d + \sum_{l=(n,d)} S_l^{orig}(v_n, v_d) - \sum_{l=(o,n)} S_l^{dest}(v_o, v_n) + (g_n - jb_n)|v_n|^2 = S_n^{gen} & \forall n \in N \\
& P_n^{min} \leq Re(S_n^{gen}) \leq P_n^{max} & \forall n \in G \\
& Q_n^{min} \leq Im(S_n^{gen}) \leq Q_n^{max} & \forall n \in G \\
& (v_n^{min})^2 \leq |v_n|^2 \leq (v_n^{max})^2 & \forall n \in N \\
& |i_l^{orig}(v)|^2 \leq (i_l^{max})^2 & \forall l \in B \\
& |i_l^{dest}(v)|^2 \leq (i_l^{max})^2 & \forall l \in B \\
& S_n^{gen} = 0 & \forall n \notin G \\
& v_n \in \mathbb{C}, S_n^{gen} \in \mathbb{C} & \forall n \in N. \qquad \text{(OPF)}
\end{aligned}
$$

Notons que nous élevons au carré les limites thermiques et les contraintes sur le module de la tension en prévision d'une formulation réelle à partir de la représentation cartésienne des nombres complexes. Ceci permet d'éviter d'utiliser une racine carrée.

Quand l'objectif est linéaire en la puissance active, il n'est pas nécessaire d'utiliser les variables de puissance S_n^{gen} pour avoir un QCQP.

2.2.2 Résolution

Le problème d'OPF présenté ci-dessus est un problème QCQP en variables complexes mais il existe d'autres modélisations. De manière générale, un problème d'OPF est un problème d'optimisation polynomial en variables complexes ($POP-\mathbb{C}$) qui est habituellement converti en variables réelles. C'est un problème d'optimisation NP-difficile [3, 4], principalement parce qu'il est non-convexe, et à ce jour il n'existe pas de méthode générique efficace pour le résoudre. Toutefois, il existe plusieurs méthodes qui permettent de calculer des solutions réalisables (i.e., des bornes supérieures) et des bornes inférieures de bonne qualité. Dans la suite de cette section, nous commençons par évoquer la conversion en variables réelles. Nous présentons ensuite les méthodes de calcul de bornes puis les méthodes d'optimisation globale qui ont été proposées pour résoudre les problèmes d'OPF.

Conversion en nombres réels

Le problème d'OPF est un problème d'optimisation polynomial naturellement formulé en variables complexes à cause de la modélisation en courant alternatif. Toutefois, il existe très peu de méthodes pour résoudre un problème d'optimisation en variables complexes et le problème est généralement converti en variables réelles. Il y a deux choix pour chaque variable complexe : la représentation cartésienne (que l'on nomme aussi représentation rectangulaire) ou la représentation polaire. En utilisant la représentation rectangulaire pour toutes les variables, le problème obtenu est un problème polynomial. L'utilisation de la représentation polaire conduit quant à elle à l'utilisation de fonctions trigonométriques. Dans la littérature autour du problème d'OPF, il est courant d'exprimer les variables de tension avec la représentation polaire et les variables de puissance avec la représentation rectangulaire. Dans tous les cas, le problème est non linéaire et non convexe.

Dans la suite de cette thèse, nous utilisons exclusivement la représentation rectangulaire pour conserver la structure polynomiale du problème d'OPF. Ainsi, chaque vecteur complexe est dédoublé : chaque variable de tension v_n s'écrit $v_n = v_{nRe} + \mathbf{j}v_{nIm}$ et chaque variable de puissance S_g^{gen} s'écrit $S_g^{gen} = P_g^{gen} + \mathbf{j}Q_g^{gen}$. Le problème (OPF) devient alors le problème (OPF-reel) suivant :

$$
\begin{aligned}
\min \quad & \sum_{g\in G} c_g(P_g^{gen}) \\
s.c. \quad & P_n^d + \sum_{l=(n,d)} P_l^{orig}(v_{Re}, v_{Im}) - \sum_{l=(o,n)} P_l^{dest}(v_{Re}, v_{Im}) + g_n(v_{nRe}^2 + v_{nIm}^2) = P_n^{gen} && \forall n \in N \\
& Q_n^d + \sum_{l=(n,d)} Q_l^{orig}(v_{Re}, v_{Im}) - \sum_{l=(o,n)} Q_l^{dest}(v_{Re}, v_{Im}) - b_n(v_{nRe}^2 + v_{nIm}^2) = Q_n^{gen} && \forall n \in N \\
& P_n^{min} \leq P_n^{gen} \leq P_n^{max} && \forall n \in G \\
& Q_n^{min} \leq Q_n^{gen} \leq Q_n^{max} && \forall n \in G \\
& (v_n^{min})^2 \leq v_{nRe}^2 + v_{nIm}^2 \leq (v_n^{max})^2 && \forall n \in N \\
& |i_l^{orig}(v_{Re}, v_{Im})|^2 \leq (i_l^{max})^2 && \forall l \in B \\
& |i_l^{dest}(v_{Re}, v_{Im})|^2 \leq (i_l^{max})^2 && \forall l \in B \\
& P_n^{gen} = 0 && \forall n \notin G \\
& Q_n^{gen} = 0 && \forall n \notin G \\
& v_{Re} \in \mathbb{R}^{|N|}, v_{Im} \in \mathbb{R}^{|N|}, P^{gen} \in \mathbb{R}^{|N|}, Q^{gen} \in \mathbb{R}^{|N|}.
\end{aligned}
$$

(OPF-reel)

dans lequel P_l^{orig}, P_l^{dest}, Q_l^{orig}, Q_l^{dest} et $|i_l^{orig}|^2$, $|i_l^{dest}|^2$ sont des quantités réelles qui ne contiennent que les produits de variables suivants (pour $l = (o, d)$) : v_{oRe}^2,v_{oIm}^2, v_{dRe}^2, v_{dIm}^2, $v_{oRe}v_{dIm}$, $v_{oIm}v_{dRe}$, $v_{oRe}v_{dRe}$, $v_{oIm}v_{dIm}$. Le problème (OPF-reel) est donc un problème quadratique avec contraintes quadratiques, sous-catégorie des problèmes d'optimisation polynomiaux.

Calcul de solutions réalisables

Il existe plusieurs méthodes pour calculer des solutions réalisables pour un problème d'OPF. Les méthodes les plus utilisées sont des méthodes de points intérieurs car elles fournissent de bons optima locaux pour l'OPF. D'autres méthodes locales ont été testées comme l'optimisation quadratique successive. Il existe aussi des méthodes non déterministes (algorithmes génétiques, colonie d'abeilles, etc). Un résumé plus exhaustif est présenté dans les études [22–24].

Calcul de bornes inférieures

Le problème de l'OPF est non convexe. Ainsi, satisfaire les conditions de premier ordre de Karush-Kuhn-Tucker ne suffit pas à prouver l'optimalité globale. Ceci est d'ailleurs confirmé dans l'article [25] qui démontre l'existence d'optima locaux pour le problème de l'OPF. Pour évaluer la qualité des solutions réalisables calculées, il est alors nécessaire de construire des relaxations convexes pour calculer des bornes inférieures.

Plusieurs types de relaxations convexes ont été proposés pour l'OPF [26–28]. Les relaxations linéaires [29, 30] sont les plus simples et celles qui sont le plus utilisées en pratique dans les études RTE car elles peuvent être résolues de manière efficace même à grande échelle. La linéarisation la plus connue mène au problème que l'on appelle DC-OPF (OPF en Courant Direct). Toutefois, ce type de linéarisation ne fournit pas des bornes inférieures de bonne qualité ni de plan de tension. Les relaxations SDP [31, 32] fournissent des bornes inférieures de bonne qualité mais leur résolution est problématique à grande échelle. Parmi les relaxations SDP, la relaxation la plus utilisée est la relaxation du rang qui peut être construite à partir de n'importe quel QCQP en nombres complexes ou réels (voir section 2.3.3). Des versions plus précises de cette relaxation ont été proposées dans [33] et [34]. Plus récemment, [35] a proposé une relaxation SDP moins coûteuse en terme de résolution mais moins précise que la relaxation du rang. Pour éviter les problèmes de résolution des relaxations SDP à grande échelle, des relaxations SOCP (Second-Order Cone Programming) ont aussi été étudiées. La relaxation SOCP classique est obtenue en relâchant la contrainte SDP de la relaxation du rang. Ainsi, la relaxation SOCP classique est de moins bonne qualité que la relaxation du rang mais elle est plus facile à résoudre notamment en grande dimension. Plusieurs méthodes ont

été proposées pour améliorer la relaxation SOCP classique [36,37]. Finalement, une relaxation quadratique convexe a été proposée dans [38]. Elle est plus précise que la relaxation SOCP classique tout en étant relativement peu coûteuse en terme de résolution. En revanche, elle n'est pas comparable à la relaxation du rang. Plusieurs méthodes ont été proposées pour améliorer cette relaxation quadratique [39, 40]. Voir la récente étude [41] pour plus de détails sur les relaxations coniques du problème d'OPF.

Certificats d'optimalité globale

Le calcul d'une borne inférieure peut être beaucoup plus coûteux que le calcul d'une solution réalisable. C'est pourquoi des certificats d'optimalité globale ont été proposés : ce sont des procédures, en général rapides et peu coûteuses, qui permettent de vérifier si une solution réalisable est un optimum global. Une procédure générique est proposée dans [42] pour les problèmes d'optimisation polynomiaux. Elle consiste à chercher la fonction objectif la plus proche de celle du problème de départ pour laquelle le point candidat est un optimum global. Cependant, cette approche est a priori coûteuse. Un certificat d'optimalité peu coûteux a été proposé pour l'OPF dans [43]. Il s'appuie sur les conditions KKT de la relaxation du rang. Plus récemment, une extension de cette condition a été proposée dans [44]. Celle-ci s'appuie sur les conditions KKT de la relaxation d'ordre d de la hiérarchie de Lasserre (reformulée avec les mineurs principaux), qui est une meilleure relaxation que la relaxation du rang si $d \geq 2$. Un certificat d'optimalité globale peut donc être intéressant si l'on a des raisons de penser que la solution réalisable calculée est globale. En revanche, si la solution candidate n'est pas un optimum global, aucune autre information n'est disponible et il faut calculer une borne inférieure pour avoir une idée de la qualité de la solution candidate.

Méthodes exactes

De manière générale, calculer une borne supérieure et une borne inférieure ne suffit pas à prouver l'optimalité globale. Il faut alors mettre en place des méthodes qui permettent de réduire l'écart entre ces deux bornes jusqu'à la précision souhaitée.

Plusieurs méthodes exactes ont été proposées pour résoudre globalement l'OPF. On distingue deux grandes classes de méthodes : des méthodes qui explorent le domaine réalisable et des méthodes de convexification convergente. Les méthodes qui explorent le domaine réalisable sont principalement des méthodes de type énumération implicite spatiale (*spatial Branch-&-Bound* en anglais) qui s'appuient sur des relaxations convexes. Dans le contexte de l'OPF, les algorithmes d'énumération implicite s'appuient sur des relaxations SDP [10], des relaxations quadratiques [9] ou encore des relaxations SOCP [37]. Plus récemment, la méthode

proposée dans [45–47] s'appuie sur un partitionnement de variables comme dans un algorithme d'énumération implicite mais explore l'espace réalisable différemment. Il existe aussi plusieurs méthodes de convexification convergente : la hiérarchie de Lasserre [7, 34, 48, 49] construit des relaxations SDP de plus en plus précises mais d'autres hiérarchies construisent des relaxations linéaires ou SOCP [50] moins coûteuses mais qui convergent plus lentement que les relaxations SDP.

A ce jour, aucune méthode n'est efficace pour résoudre toutes les instances d'OPF. En particulier, la plupart de ces méthodes sont trop coûteuses pour des réseaux de grande taille.

2.2.3 Variantes

Il existe de nombreuses variantes du problème de l'OPF. Ces variantes peuvent par exemple traiter l'incertitude sur la demande ou sur la production (e.g. via un problème d'OPF robuste [51]), assurer la sécurité d'approvisionnement en anticipant des défauts sur le réseau (des « N-1 », voir problème décrit ci-dessous) ou encore permettre de se prémunir des problèmes de tension en prenant en compte des moyens d'action sur la tension (voir problème OPF réactif). Dans cette thèse, deux variantes du problème de l'OPF nous intéressent. Elles sont décrites ci-dessous. Ces variantes étant des Problèmes Non Linéaires en variables Mixtes (MINLP) non convexes, nous proposons ensuite un bref état-de-l'art des techniques de résolution de ce type de problème.

OPF avec contraintes « N-1 »

Pour faire face aux défauts du réseau, des contraintes « N-1 » peuvent être ajoutées dans un problème d'OPF : elles assurent la réalisabilité d'une solution en cas de perte d'un équipement (par exemple une ligne ou un générateur). Le problème résultant s'appelle le PSCOPF (OPF Préventif avec Contraintes de Sécurité). Une version de ce problème a été proposée dans la phase Beta de la compétition ARPA-E Grid Optimization [52]. Le réseau est décrit avec une liste de défauts possibles (perte d'une ligne, d'un générateur ou d'un transformateur) et le problème à résoudre est alors une combinaison de plusieurs problèmes d'OPF couplés. Ce problème est présenté plus en détail dans la section 3.4.

OPF réactif

Pour modéliser plus finement le fonctionnement du réseau de transport d'électricité, il est possible de prendre en compte différentes actions : mettre en service ou hors service des MCS, régler la prise des transformateurs, faire des changements topologiques, etc. Intégrer ces pos-

sibilités dans le problème de l'OPF implique l'utilisation de variables binaires et/ou entières, ce qui complexifie encore le problème. Le problème ROPF (OPF Réactif) se concentre sur deux moyens d'action qui permettent de contrôler la tension : mise en service/hors service des MCS et changement du ratio des transformateurs. Ce problème est présenté plus en détail dans la section 5.2.

L'optimisation MINLP non convexe

Les variantes de l'OPF comme celles présentées ci-dessus sont généralement des Problèmes Non Linéaires en variables Mixtes (MINLP) dont la relaxation continue est non convexe. On les appelle alors des problèmes MINLP non convexes. Ce sont des problèmes NP-difficiles pour lesquels il est parfois difficile en pratique de trouver une solution réalisable [53]. Il existe de nombreuses heuristiques pour ces problèmes : principalement des adaptations d'algorithmes pour les problèmes MINLP convexes [54] ou pour les problèmes linéaires en variables mixtes [55–57]. Il existe aussi des heuristiques qui s'appuient sur une reformulation avec des contraintes d'équilibre [58]. Par ailleurs, des méthodes exactes ont été proposées. Les plus connues sont les méthodes d'énumération implicite spatiale (*spatial Branch-and-Bound* en anglais) mais il existe d'autres méthodes comme les techniques de conversion en un problème linéaire en variables mixtes [53]. Dans le cas particulier des problèmes non convexes quadratiques ou polynomiaux en variables mixtes, toutes les méthodes présentées pour l'OPF sont envisageables (linéarisation, utilisation de relaxations convexes comme les relaxations SDP, etc).

2.3 L'optimisation semi-définie positive

L'optimisation semi-définie positive ou SDP est un domaine de l'optimisation convexe qui s'est particulièrement développé à la fin des années 90 [11]. L'optimisation SDP est une généralisation de l'optimisation linéaire : dans un problème SDP, les variables sont des matrices semi-définie positives, généralisation des variables scalaires positives des problèmes d'optimisation linéaire. Les contraintes et l'objectif d'un problème SDP sont linéaires. Par ailleurs, l'optimisation SDP est un cas particulier de l'optimisation conique, qui consiste à minimiser une fonction linéaire sur l'intersection d'un cône convexe fermé et d'un sous-espace affine. Dans la suite de cette section, nous détaillons la formulation d'un problème SDP et de son dual puis nous présentons les principales méthodes de résolution. Finalement, nous exposons les applications de l'optimisation SDP à l'optimisation polynomiale.

2.3.1 Définition du problème primal et du problème dual

Un problème SDP s'écrit de la façon suivante :

$$\begin{cases} \min\limits_{X} : C \cdot X \\ s.c.\ A_p \cdot X = b_p\ \forall p = 1..m \\ X \succeq 0, \end{cases} \qquad \text{(SDPprimal)}$$

où C, X et A_p sont des matrices symétriques réelles de taille $n \times n$ ($C, X, A_p \in \mathbb{S}^{n\times n}$) et b est un vecteur réel de taille m ($b \in \mathbb{R}^m$). On considère que les matrices A_p sont linéairement indépendantes dans $\mathbb{S}^n$.

Le dual d'un problème SDP est aussi un problème SDP. Le dual du problème SDP ci-dessus s'écrit :

$$\begin{cases} \max\limits_{y,Z} : b^T y \\ s.c.\ Z + \sum\limits_{p=1}^{m} A_p y_p = C \\ Z \succeq 0, \end{cases} \qquad \text{(SDPdual)}$$

où $Z \in \mathbb{S}^n$ et $y \in \mathbb{R}^m$.

En optimisation linéaire, le problème primal et le problème dual ont même valeur lorsqu'ils sont réalisables (dualité forte). En optimisation SDP, ce n'est pas toujours le cas : il peut y avoir un saut de dualité, c'est-à-dire que le problème primal et le problème dual n'ont pas nécessairement la même valeur. Le saut de dualité est nul à la condition que l'intérieur strict d'un des deux problèmes ne soit pas vide (qualification de Slater).

Les problèmes SDP sont généralement définis en nombres réels mais des problèmes SDP en nombres complexes peuvent également être définis avec des matrices hermitiennes. Nous conseillons les manuels [59, 60] pour plus de détail sur la théorie et les algorithmes pour l'optimisation SDP.

2.3.2 Résolution

Méthodes de points intérieurs

Les problèmes SDP peuvent être résolus en temps polynomial par des algorithmes de points intérieurs [13]. Ces algorithmes sont très efficaces pour des problèmes de taille moyenne mais ne se comportent pas bien à grande échelle à cause de la matrice hessienne du Lagrangien qu'il

faut construire, stocker en mémoire et factoriser. Des améliorations sont possibles en utilisant le caractère creux des problèmes [14, 61] mais la résolution de gros problèmes SDP par les algorithmes de points intérieurs reste un défi. Les deux paragraphes suivants présentent les autres méthodes proposées pour résoudre des problèmes SDP creux.

Décomposition en cliques maximales

La décomposition en cliques maximales permet de reformuler un problème SDP en exploitant le caractère creux. Ce problème reformulé est généralement résolu par une méthode de points intérieurs qui est alors beaucoup plus efficace. Il peut aussi être résolu par une autre méthode comme les méthodes de premier ordre présentées dans le paragraphe suivant.

La décomposition en cliques repose sur le théorème de complétion matricielle [15, 16, 62]. Celui-ci permet de remplacer la contrainte SDP sur la matrice X par plusieurs contraintes SDP sur des sous-matrices plus petites. En contrepartie, des contraintes liantes sont ajoutées entre ces sous-matrices. Elles permettent donc de reformuler (SDPprimal) de la façon suivante :

$$\begin{cases} \min & \sum_{i=1}^{r} C^i \cdot X^i \\ s.c. & \sum_{i=1}^{r} A_p^i \cdot X^i = b_p \ \forall p = 1..m \\ & X^i_{\sigma_i[k],\sigma_i[l]} = X^{i'}_{\sigma_{i'}[k],\sigma_{i'}[l]} \ \forall\ k,l \in K_i \cap K_{i'} \ \forall \text{ cliques } K_i, K_{i'} \text{ [contraintes liantes]} \\ & X^i \succeq 0 \ \forall i = 1..r \end{cases}$$

(SDPdécomposé)

avec r le nombre de cliques maximales (les cliques maximales étant notées K_i), les sous-matrices variables X^i correspondant aux cliques maximales (de taille $|K_i|$) et les matrices C^i et A_p^i construites à partir des matrices des données C et A_p et adaptées aux cliques maximales. Le dictionnaire σ_i retourne l'index d'un sommet $k \in K_i$ dans la sous-matrice X^i. Par exemple, si la clique K_1 contient les sommets 2, 3 et 5 alors $\sigma_1[2] = 1$, $\sigma_1[3] = 2$ et $\sigma_1[5] = 3$.

Plus précisément, soit A la matrice de parcimonie d'un problème SDP, i.e., la matrice de taille $n \times n$ qui représente les coefficients non nuls dans la matrice objectif C et les matrices de contraintes A_p. Ainsi $A[i,j] = 0$ si et seulement si $C[i,j] = 0$ et $A_p[i,j] = 0\ \forall p = 1..m$; sinon $A[i,j] = 1$ (ou tout autre coefficient non nul). On note G^A le graphe associé à A dans lequel il y a une arête entre i et j si $A[i,j] \neq 0$. Le théorème de complétion matricielle peut être appliqué si et seulement si G^A est cordal. Un graphe cordal est un graphe qui n'a pas de cycle

de longueur 4 ou plus comme sous-graphe induit. Autrement dit, chaque cycle de longueur 4 ou plus d'un graphe cordal possède une corde, i.e., une arête qui relie deux sommets non adjacents. Une clique est un ensemble de sommets qui sont tous adjacents deux à deux. Une clique maximale est une clique qui n'est contenue dans aucune autre clique.

Il y a trois étapes pour obtenir une décomposition en cliques maximales. D'abord, il faut calculer une extension cordale H du graphe G^A, c'est-à-dire ajouter des arêtes à G^A pour obtenir un graphe cordal. Par exemple, la figure 2.4 montre une extension possible pour le graphe représenté dans la figure 2.3. Ensuite, il faut calculer la liste des cliques maximales dans H, notée L. Pour l'exemple donné dans la figure 2.4, on obtient 7 cliques maximales : $L = \{\{2, 8\}, \{7, 8, 9\}, \{6, 7, 9\}, \{5, 6, 9\}, \{3, 6\}, \{4, 5, 9\}, \{1, 4\}\}$. À partir de la liste des cliques maximales L, on peut définir un graphe complet W dans lequel chaque sommet K_i représente une clique maximale et où le poids de chaque arête $e = (K_i, K_j)$ est défini par le nombre de nœuds en commun entre les deux cliques K_i et K_j. La troisième étape consiste à calculer un arbre couvrant de poids maximum pour W, noté T et appelé arbre de cliques. Cet arbre de cliques permet d'écrire les contraintes liantes entre les cliques de manière efficace. La figure 2.5 représente un arbre de cliques pour les cliques de l'exemple donné dans la figure 2.4. Tous les graphes ont été obtenus avec le module GraphPlot.jl.

Cette procédure assez simple soulève toutefois de nombreuses questions. En effet, il y a plusieurs possibilités pour calculer une extension cordale et la décomposition en cliques dépend de l'extension cordale. En pratique, on ne sait pas ce qu'est une "bonne" décomposition en cliques et on cherche à minimiser le nombre d'arêtes ajoutées dans l'extension cordale. Trouver une extension cordale minimale est NP-complet [63] mais il existe plusieurs heuristiques efficaces [64]. Bergman et al. ont récemment proposé un modèle exact avec un nombre de contraintes exponentiel [65] mais ce modèle n'est pas utilisable pour des gros graphes. L'une des heuristiques les plus utilisées en pratique s'appuie sur une factorisation de Cholesky précédée d'une permutation sur les lignes et les colonnes suivant un ordre des degrés minimum ou approximativement mimimum [66]. L'ordre choisi a un impact significatif sur le nombre d'arêtes ajoutées. Une fois que l'extension cordale est calculée, la liste des cliques maximales est unique et peut être déterminée en temps polynomial car le graphe est cordal [67]. Enfin, l'algorithme exact le plus utilisé pour calculer l'arbre de cliques est l'algorithme de Prim [68]. L'arbre de cliques n'est pas forcément unique et d'autres algorithmes peuvent calculer un arbre de cliques différent. Cependant, les décompositions issues d'arbres de cliques différents produisent des décompositions similaires car la seule différence se trouve dans le nombre de contraintes liantes. Or ce nombre ne peut pas varier beaucoup d'un arbre à l'autre car tous les arbres possibles ont le même poids total. In fine, l'étape cruciale pour définir la décomposition en cliques est le calcul de l'extension cordale.

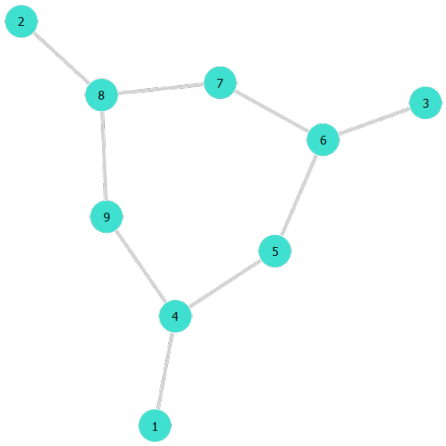

Figure 2.3 Exemple de graphe G^A - Graphe non cordal car il existe un cycle induit de 6 nœuds (4-5-6-7-8-9).

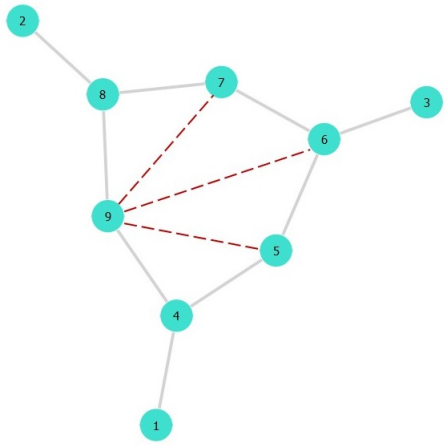

Figure 2.4 Exemple d'extension cordale H pour G^A

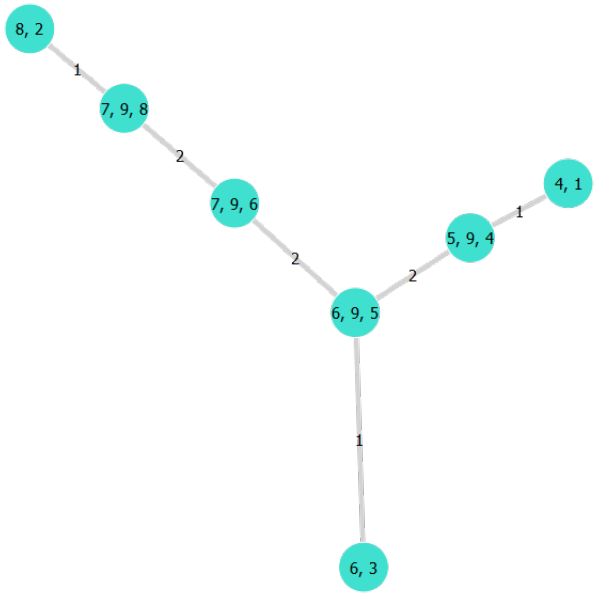

Figure 2.5 Exemple d'arbre de cliques T pour les cliques maximales de H

Récemment, [17] a montré qu'une méthode classique de décomposition en cliques maximales pouvait être suivie d'une procédure de réduction faciale pour améliorer les performances de résolution. La réduction faciale consiste aussi à reformuler un problème SDP pour améliorer la résolution numérique. Cette technique a surtout été appliquée aux problèmes SDP dégénérés (pour lesquels la condition de Slater n'est pas vérifiée) afin de les reformuler en des problèmes avec un point de Slater [69, 70].

Méthodes de premier ordre

Des méthodes de premier ordre ont été proposées pour résoudre les problèmes SDP à la place des algorithmes de points intérieurs. Ces méthodes sont en effet moins coûteuses que les points intérieurs mais elles peuvent être moins précises et/ou connaître des problèmes de convergence. Elles peuvent donc être intéressantes à grande échelle si une résolution à précision modeste est acceptable. On distingue principalement deux grandes familles de méthodes : les algorithmes de type Dantzig-Wolfe ou méthode des faisceaux [19, 71] et les méthodes de type ADMM (Méthodes des Multiplicateurs à Direction Alternée) [18, 72–75] qui connaissent un grand succès en ce moment et qui ont été testées sur les relaxations SDP de l'OPF. D'autres méthodes de premier ordre légèrement différentes ont été proposées pour résoudre les relaxations SDP de l'OPF [20]. Cependant, la convergence des méthodes de type ADMM dépend fortement du choix du paramètre de pénalisation du lagrangien augmenté et il n'existe pas de règle générique qui permette de trouver une bonne valeur pour ce paramètre. Un algorithme de type Benders a aussi été proposé dans un cas particulier [76].

2.3.3 Application à l'optimisation polynomiale

L'optimisation SDP est intéressante car beaucoup de problèmes convexes ont une formulation SDP, ce qui prouve qu'ils peuvent être résolus en temps polynomial. Par ailleurs, il existe des relaxations SDP très précises pour certains problèmes non convexes, comme les problèmes polynomiaux. Dans les paragraphes suivants, nous présentons les différentes relaxations SDP qui peuvent être définies pour un problème d'optimisation polynomial.

Définition d'un problème d'optimisation polynomial

Une fonction polynomiale est une fonction f de $\mathbb{R}^n$ dans $\mathbb{R}$ telle que $f(x) = \sum_{\alpha} c_\alpha \prod_{k=1}^{n} z_k^{\alpha_k}$ avec α des vecteurs de nombres entiers de taille n et c_α un coefficient réel défini pour chaque vecteur α.

Un problème d'optimisation polynomial en variables réelles s'écrit de la façon suivante :

$$\begin{array}{ll} \min & f(x) \\ \text{s.c.} & g_i(x) \geq 0 \quad \forall i = 1..m \\ & x \in \mathbb{R}^n \end{array} \tag{POP}$$

avec f et g_i des fonctions polynomiales.

Le même type de problème peut être défini en variables complexes avec des fonctions polynomiales de $\mathbb{C}^n$ dans $\mathbb{R}$.

La relaxation du rang

La relaxation du rang est la relaxation SDP la plus communément utilisée pour les problèmes QCQP. Or, un problème d'optimisation polynomial en variables réelles ou complexes peut toujours être reformulé sous forme de QCQP en introduisant de nouvelles variables. Soit (2.14) un QCQP non convexe :

$$\begin{cases} \min & v^H Q_0 v \\ s.c. & v^H Q_p v \leq a_p \ \forall p = 1..m \\ & v \in \mathbb{C}^n, \end{cases} \tag{2.14}$$

avec les matrices complexes Q_i de taille $n \times n$ et le vecteur complexe a de taille m.

La matrice hermitienne $W = vv^H$ est introduite pour réécrire le problème (2.14). La contrainte $W = vv^H$ équivaut à $W \succeq 0$ et $\mathsf{rang}(W) = 1$. La seule contrainte non convexe est alors la contrainte de rang. Pour obtenir une relaxation SDP, la contrainte de rang est supprimée comme décrit dans (2.15).

$$\begin{cases} \min & Q_0 \cdot W \\ s.c. & Q_p \cdot W \leq a_p \ \forall p = 1..m \\ & W = vv^H \end{cases} \Rightarrow \begin{cases} \min & Q_0 \cdot W \\ s.c. & Q_p \cdot W \leq a_p \ \forall p = 1..m \\ & \cancel{\mathsf{rang}(W) = 1} \\ & W \succeq 0 \end{cases} \tag{2.15}$$

Le problème résultant est formulé en variables complexes mais il peut être réécrit sous forme de problème SDP en variables réelles en utilisant la représentation rectangulaire et une matrice X de taille $2n$.

La relaxation du rang peut aussi être définie pour des problèmes combinatoires. L'application la plus connue concerne le problème de coupe maximale [77].

La relaxation du rang correspond à la relaxation d'ordre 1 de la hiérarchie de Lasserre présentée ci-dessous.

La hiérarchie de Lasserre

Introduite en 2001, la hiérarchie de Lasserre [78] vise à résoudre des problèmes d'optimisation polynomiaux de manière exacte grâce à l'optimisation SDP. Plus précisément, elle fournit une suite de relaxations SDP qui sont de plus en plus précises lorsque l'ordre de la hiérarchie augmente. Cette suite étant convergente, la hiérarchie de Lasserre est une méthode permettant de prouver l'optimalité globale. Toutefois, elle est limitée en pratique car les relaxations SDP deviennent rapidement de grande taille lorsque l'on augmente l'ordre. La hiérarchie de Lasserre peut être définie sur des problèmes d'optimisation polynomiaux en variables réelles ou complexes [7]. Nous présentons cette hiérarchie en variables réelles par souci de simplicité.

La relaxation SDP de la hiérarchie de Lasserre d'ordre d s'appuie sur la matrice des moments d'ordre d, notée $\mathcal{M}_d$, qui est définie à partir du vecteur des monômes d'ordre inférieur ou égal à d. Celui-ci s'écrit :

$$x_d = \begin{pmatrix} 1 & x_1 & x_2 & \dots & x_{n-1}x_n^{d-1} & x_n^d \end{pmatrix}^T.$$

La matrice des moments représente tous les produits des monômes : $\mathcal{M}_d = x_d x_d^T$. Par exemple, les matrices des moments d'ordre 1 et 2 sont explicitées dans le tableau 2.1 pour un problème avec 2 variables réelles.

Soit d l'ordre de la relaxation. La matrice des moments d'ordre d $\mathcal{M}_d$ est introduite pour reformuler le problème (POP). Chaque contrainte $g_i(x)$ est multipliée par une matrice des moments d'ordre $d - d_i$ tel que tous les coefficients issus du produit entre la contrainte et la matrice des moments soient de degré inférieur ou égal à $2d$ (donc représentables par la matrice d'ordre d). Ceci donne le problème suivant :

Tableau 2.1 Matrice des moments d'ordre 1 et 2 pour un problème d'optimisation polynomial avec 2 variables réelles

d	1	2
x_d	$\begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix}$	$\begin{pmatrix} 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \end{pmatrix}$
$\mathcal{M}_d$	$\begin{pmatrix} 1 & x_1 & x_2 \\ x_1 & x_1^2 & x_1x_2 \\ x_2 & x_2x_1 & x_2^2 \end{pmatrix}$	$\begin{pmatrix} 1 & x_1 & x_2 & x_1x_2 & x_1^2 & x_2^2 \\ x_1 & x_1^2 & x_1x_2 & x_1^2x_2 & x_1^3 & x_1x_2^2 \\ x_2 & x_2x_1 & x_2^2 & x_2^2x_1 & x_2x_1^2 & x_2^3 \\ x_1x_2 & x_1^2x_2 & x_1x_2^2 & x_1^2x_2^2 & x_1^3x_2 & x_1x_2^3 \\ x_1^2 & x_1^3 & x_1^2x_2 & x_1^3x_2 & x_1^4 & x_1^2x_2^2 \\ x_2^2 & x_2^2x_1 & x_2^3 & x_2^3x_1 & x_2^2x_1^2 & x_2^4 \end{pmatrix}$

$$\begin{aligned} \min \quad & f(x) \\ \text{s.c.} \quad & g_i(x)\mathcal{M}_{d-d_i} \succeq 0 \quad \forall i = 1..m \\ & \mathcal{M}_d = x_d x_d^T. \end{aligned} \tag{2.16}$$

Soit y le vecteur représentant les coefficients de la matrice $\mathcal{M}_d$, que l'on appelle aussi les moments. Alors on peut réécrire la fonction objectif $f(x)$ en fonction des moments, ce que l'on note $\mathcal{L}_y(f)$. De même, on peut réécrire chaque contrainte $g_i(x)\mathcal{M}_{d-d_i} \succeq 0$ en fonction des moments, ce que l'on note $\mathcal{M}_{d-d_i}(g_i y)$. Enfin, la contrainte liant $\mathcal{M}_d$ et x_d peut être remplacée par $\mathcal{M}_d \succeq 0$ et $\mathsf{rang}(\mathcal{M}_d) = 1$. Ainsi le problème 2.16 peut être reformulé en 2.17. Finalement, il suffit de supprimer la contrainte de rang pour obtenir la relaxation SDP d'ordre d.

$$\begin{aligned} \min \quad & \mathcal{L}_y(f) \\ \text{s.c.} \quad & \mathcal{M}_{d-d_i}(g_i y) \succeq 0 \qquad\qquad \forall i = 1..m \\ & \mathcal{M}_d = x_d x_d^T \Leftrightarrow \begin{cases} \mathcal{M}_d \succeq 0 \\ \cancel{\mathsf{rang}(\mathcal{M}_d) = 1} \end{cases} \end{aligned} \tag{2.17}$$

La dualité forte a été prouvée pour la hiérarchie de Lasserre dans [79] à la seule condition d'avoir une contrainte de boule (il existe $R \in \mathbb{R}^+$ tel que $||x||^2 \leq R^2$). Pour le problème de l'OPF, une contrainte de boule peut être introduite artificiellement en sommant les contraintes sur le module de la tension. La condition de Slater est donc vérifiée pour les relaxations SDP de l'OPF provenant de la hiérarchie de Lasserre.

Il est très coûteux de passer d'un ordre à l'autre car beaucoup de moments sont ajoutés, ce qui augmente significativement la taille des matrices SDP. Pour faire face à ce problème, des approches de type hiérarchie de Lasserre partielle ont été proposées dans [7, 80]. Ces approches reposent sur deux idées. La première idée est d'exploiter le creux en utilisant des décompositions en cliques maximales. La deuxième idée consiste à augmenter l'ordre sur certaines contraintes seulement pour améliorer la précision d'une relaxation tout en contrôlant la taille des matrices SDP. Récemment, [81] a proposé une généralisation de ces travaux pour les problèmes d'optimisation polynomiaux génériques (en variables réelles).

Par ailleurs, d'autres méthodes ont été proposées pour exploiter le caractère creux dans des contextes spécifiques [82].

2.4 Le langage Julia

Le langage Julia [83] a été créé par Jeff Bezanson, Stefan Karpinski, Viral B. Shah et Alan Edelman (des chercheurs du MIT) en 2009. C'est un langage de programmation haut niveau pour le calcul scientifique. L'intérêt de ce langage est d'être à la fois pratique et performant : la syntaxe est proche de celle de Matlab et les performances comparables à celles du C. Par ailleurs, c'est un langage open-source, même s'il existe aujourd'hui une société JuliaComputing qui vend certains services.

Le module JuMP.jl [84] est le module de programmation mathématique en Julia. Il permet de modéliser un certain nombre de problèmes d'optimisation (linéaires, avec des variables binaires, non linéaires, SOCP, SDP) et d'utiliser plusieurs solveurs, libres ou commerciaux. Il se rapproche donc d'un langage de modélisation comme AMPL [85] mais avec plus de possibilités.

Le langage Julia est approprié dans le contexte de ce projet de recherche car il permet de tester rapidement des nouvelles approches (simplicité de la syntaxe) tout en assurant des performances raisonnables, ce qui est nécessaire puisque nous travaillons avec des réseaux de grande taille. De plus, le module JuMP.jl est très intéressant pour nous puisqu'il permet de modéliser des problèmes SDP et de tester plusieurs solveurs SDP.

CHAPITRE 3 MATHPROGCOMPLEX.JL : UN MODULE JULIA POUR L'OPTIMISATION POLYNOMIALE EN VARIABLES COMPLEXES

Dans ce projet de recherche nous cherchons à améliorer la résolution des relaxations SDP pour les problèmes d'OPF. Ce chapitre présente les outils développés afin de modéliser tous les problèmes d'OPF décrits dans la section 2.2. Il est organisé de la façon suivante : après avoir exposé les motivations qui nous poussent à concevoir de nouveaux outils, nous présentons les principaux outils développés et un exemple de problème d'OPF facilement traité grâce à nos outils.

3.1 Motivations

De manière générale, les problèmes d'OPF sont des problèmes d'optimisation non convexes avec des variables complexes. Lorsqu'ils ne contiennent que des variables continues, ils font partie de la classe des problèmes d'optimisation polynomiaux en variables complexes ($POP-\mathbb{C}$) mis en lumière par Josz et Molzahn dans [7]. Nous proposons d'étendre cette classe aux problèmes mixtes ($MIPOP-\mathbb{C}$) afin d'inclure les problèmes d'OPF avec des variables entières ou binaires.

Les problèmes d'optimisation polynomiaux sont des problèmes dans lesquels l'objectif et les contraintes sont décrits par des polynômes multivariés. Pour un problème d'optimisation polynomial avec des variables complexes ($POP-\mathbb{C}$), les polynômes sont des fonctions de $\mathbb{C}^n$ dans $\mathbb{R}$, c'est-à-dire qu'ils ont des coefficients et des variables complexes mais ils sont à valeur réelle. Plus précisément, un tel polynôme f prend en entrée un vecteur de complexes z et retourne une valeur réelle $f(z) = \sum_{\alpha,\beta} p_{\alpha,\beta} z^\alpha \overline{z}^\beta$ calculée à partir de coefficients complexes $p_{\alpha,\beta}$ et de puissances de z et de son conjugué $\overline{z}$. Par exemple, le polynôme $g(z) = z_1\overline{z_1} - 0.5\mathbf{j}z_2 + 0.5\mathbf{j}\overline{z_2}$ est un polynôme complexe à valeur réelle. Les problèmes ($POP-\mathbb{C}$) peuvent être étendus aux problèmes ($POP-\mathbb{C}$) en variables mixtes dans lesquels certaines variables peuvent être réelles et entières. Un tel problème, noté ($MIPOP-\mathbb{C}$), s'écrit de la manière suivante :

$$
\begin{aligned}
\min_{z} \quad & \sum_{\alpha,\beta} p_{0,\alpha,\beta} z^\alpha \overline{z}^\beta \\
\text{s.c.} \quad & \sum_{\alpha,\beta} p_{i,\alpha,\beta} z^\alpha \overline{z}^\beta \geq 0 \ \forall i = 1...m \qquad (MIPOP-\mathbb{C}) \\
& z \in \mathbb{C}^n, z_k \in \mathbb{N} \ \forall k \in K \subset \{1...n\}.
\end{aligned}
$$

dans lequel $\overline{z}$ représente le conjugué de z et $z^\alpha = \prod_{i=1}^{n} z_i^{\alpha_i}$. Les sommes sur $\alpha, \beta \in \mathbb{N}^n$ sont finies et les coefficients $p_{i,\alpha,\beta}$ appartiennent à $\mathbb{C}$.

Malgré l'importance croissante de cette catégorie de problème, nous avons identifié un manque d'outil dédié. En effet, même si certains outils comme le module Matlab CVX [86, 87] ou le module Julia Convex.jl [88] supportent les nombres complexes, ils ne peuvent pas traiter des problèmes non convexes comme l'OPF. D'autres modules pour les systèmes électriques exploitent la structure complexe : MATPOWER [21] dans la procédure d'autodifférentiation ou le module Julia PowerModels.jl [89]. Cependant, ces deux modules définissent les problèmes en variables réelles. Nous avons donc implémenté un module Julia afin de répondre au manque d'outil dédié aux problèmes ($MIPOP-\mathbb{C}$) et de modéliser facilement l'OPF en variables complexes.

3.2 MathProgComplex.jl : principes et exemples

Notre module MathProgComplex.jl permet de représenter les problèmes ($MIPOP-\mathbb{C}$) d'une manière générique et de tester des méthodes pour l'optimisation en variables complexes [7,90]. Ce module fournit des structures et des méthodes adaptées aux problèmes ($MIPOP-\mathbb{C}$). En particulier, les opérations de base (addition, multiplication, conjugaison, module) sont implémentées pour les polynômes complexes. Ainsi, les problèmes d'optimisation peuvent être implémentés de manière très intuitive. Par exemple, le polynôme $(1+4\mathbf{j})x^2\overline{y}^3+3xy+bx$ avec x, y des variables complexes et b une variable binaire peut être implémenté de la manière suivante avec MathProgComplex.jl :

```
x = Variable("x", Complex)
y = Variable("y", Complex)
b = Variable("b", Bool)
p = (1+4*im)*x^2*conj(y)^3+3*x*y+b*x
```

où $conj(z)$ représente le conjugué de $z \in \mathbb{C}$.

Par ailleurs, une fonction d'évaluation est implémentée. Elle permet d'évaluer un polynôme en un point donné, e.g., le polynôme défini ci-dessus au point $(1+2\mathbf{j}, 3\mathbf{j}, 0)$:

```
pt = Point([x,y,b],[1+2*im,3*im,0])
evaluate(p,pt)
>-145+28im.
```

Les problèmes ($MIPOP-\mathbb{C}$) sont aussi implémentés de manière très intuitive. Par exemple, le problème (P) :

$$\begin{array}{ll} \min\limits_{x} & \dfrac{1}{2}(x+\overline{x}-\mathbf{j}(x-\overline{x})) \\ \text{s.t.} & |x|^2 = 1 \\ & x \in \mathbb{C} \end{array} \tag{P}$$

s'écrit :

```
p = Problem()
x = Variable("x", Complex)
set_objective!(pb,0.5*(x+conj(x)-im*(x-conj(x)))
add_constraint!(pb,abs2(x) == 1)
```

avec $abs2(z)$ le module au carré de $z \in \mathbb{C}$.

Pour résoudre un problème complexe comme (P), il faut le convertir en un problème réel. Notre module fournit la fonction $pb\_cplx2real(\cdot)$ qui permet de convertir les problèmes en utilisant la représentation rectangulaire (ce qui conserve une structure polynomiale). Pour le problème (P), ($pb\_cplx2real(p)$) retourne le problème suivant :

$$\begin{array}{ll} \min\limits_{x_{Re},x_{Im}} & x_{Re}+x_{Im} \\ \text{s.t.} & x_{Re}^2+x_{Im}^2 = 1 \\ & x_{Re}, x_{Im} \in \mathbb{R} \end{array} \tag{Preal}$$

Finalement, le problème obtenu peut soit être converti en modèle JuMP [84] (un module optimisation du langage Julia) soit être exporté via un fichier texte. Cet export en fichier texte permet d'utiliser le langage souhaité pour la résolution (par exemple, un modèle AMPL [85] peut être créé).

Grâce à toutes ses fonctionnalités, le module MathProgComplex.jl permet de modéliser facilement n'importe quel problème ($MIPOP-\mathbb{C}$) et de le convertir automatiquement en variables réelles au moment de la résolution. Plus particulièrement, ce module permet de construire facilement des problèmes d'OPF.

3.3 Application à l'OPF

Le module MathProgComplex.jl est un module générique pour les problèmes ($MIPOP - \mathbb{C}$) mais la principale application qui nous intéresse est l'OPF. A partir de MathProgComplex.jl, nous avons donc conçu un cadre générique qui permet de modéliser toutes les variantes d'OPF. Ce besoin de flexibilité est en effet très important puisque les problèmes d'optimisation deviennent de plus en plus compliqués dans le contexte de la transition énergétique (plus d'incertitude sur la production, de plus en plus de problèmes de tension haute, etc).

Nous avons créé une structure flexible dans laquelle l'utilisateur définit lui-même le réseau et ses éléments. Nous définissons un réseau comme un graphe, c'est-à-dire un ensemble de nœuds et un ensemble d'arêtes. A chaque élément du réseau (nœud ou arête), un ou plusieurs éléments peuvent être associés, ce qui permet une grande flexibilité de modélisation. L'utilisateur doit spécifier le graphe du réseau et un ensemble d'éléments pour chaque nœud et pour chaque arête. Pour clarifier, l'utilisateur peut définir un élément associé à un nœud en spécifiant :

- les variables liées à l'élément ;
- sa contribution dans le bilan de puissance ;
- les contraintes engendrées par cet élément ;
- sa contribution dans la fonction coût.

Cette manière de définir les éléments permet d'écrire le bilan de puissance à chaque nœud de manière automatique puisqu'il suffit de parcourir tous les éléments et d'ajouter leur contribution.

Par exemple, pour définir le problème (OPF) décrit dans la section 2, 4 éléments sont possiblement liés à un nœud dont 1 qui est associé à chaque nœud. Ils sont résumés dans le tableau 3.1. Le premier élément est la tension qui définit une variable complexe avec des bornes sur son module. Une tension n'engendre pas de contribution dans le bilan de puissance, ni de coût. Chaque nœud possède une tension associée. Les trois éléments suivants ne concernent que certains nœuds. Le deuxième élément est la charge, soit la demande électrique, qui contribue seulement dans le bilan de puissance. Le troisième élément est le MCS (Moyen de Compensation Statique) qui intervient seulement dans le bilan de puissance en fonction de la tension au nœud. Un MCS est un moyen de contrôle de la tension et n'est présent qu'en certains nœuds du réseau. Finalement le dernier élément est la production qui définit une variable de puissance complexe avec des bornes sur sa partie réelle et sur sa partie imaginaire et une contribution dans le bilan de puissance. Cette contribution correspond à l'opposé de la partie gauche de la contrainte de bilan dans (OPF). L'élément de production est le seul élément qui a une contribution dans la fonction coût : il y a un coût quadratique

ou linéaire qui dépend de la puissance active.

De manière similaire, un élément d'arête est défini en spécifiant :

— les variables liées à l'élément ;
— sa puissance à la destination de l'arête ;
— sa puissance à l'origine de l'arête ;
— les contraintes associées à cet élément ;
— sa contribution dans la fonction coût.

Pour définir le problème (OPF), il suffit d'un seul élément associé à chaque arête : la ligne de transmission en Π décrite dans le tableau 3.2. Cet élément implique deux variables de tension : une à l'origine de l'arête et une à l'extrémité de l'arête. Les puissances à l'origine et à la destination sont définies dans les équations (2.5) et (2.6). Finalement, des contraintes thermiques en intensité sont induites pour un tel élément (pour rappel, l'intensité est liée à la tension comme défini dans (2.2)).

3.4 Validation sur un problème d'OPF avec contraintes de sécurité

Nous avons travaillé sur le problème d'Optimisation des Flux de Puissance Préventif avec Contraintes de Sécurité (PSCOPF) dans le but de valider notre module MathProgComplex.jl et la structure générique définie pour les problèmes de type OPF. En effet, nous voulions que la structure générique définie puisse modéliser un problème aussi compliqué que le problème PSCOPF. Par ailleurs, nous avions des instances avec leurs solutions à disposition : la phase Beta de la compétition Grid Optimization (GO) [52] proposait de s'entraîner sur des instances de PSCOPF et de tester nos résultats grâce à une plateforme de test. Des fichiers solution étaient aussi donnés avec les instances.

Tableau 3.1 Éléments associés à un nœud

Élément	Variables	Puissance	Contraintes	Coût
Tension	V_n	-	$\mathbf{V_n^{min}} \leq \lvert V_n \rvert \leq \mathbf{V_n^{max}}$	-
Charge	-	$\mathbf{S_n^{load}}$	-	-
MCS	-	$(\mathbf{g_n} - \mathbf{jb_n})\lvert v_n \rvert^2$	-	-
Production	S_n^{gen}	$-S_n^{gen}$	$\mathbf{P_g^{min}} \leq Re(S_g^{gen}) \leq \mathbf{P_g^{max}}$ $\mathbf{Q_g^{min}} \leq Im(S_g^{gen}) \leq \mathbf{Q_g^{max}}$	$c_n(Re(S_n^{gen}))$

Tableau 3.2 Éléments associés à une arête

Élément	Variables	Puissance à l'origine	Puissance à la destination	Contraintes	Coût
Ligne de transport Π	$V_{o(b)}$,$V_{d(b)}$	$S_b^{orig}(V)$	$S_b^{dest}(V)$	$\lvert i_b^{orig}(V)\rvert \leq \mathbf{i_l^{max}}$ $\lvert i_b^{dest}(V)\rvert \leq \mathbf{i_l^{max}}$	-

3.4.1 Présentation du problème

Le problème PSCOPF est un problème de type OPF qui anticipe des défauts sur le réseau et qui détermine un plan de production (et de tension) pour l'état normal du réseau et pour ces états défaillants (cas « N-1 »). Un cas « N-1 » est défini par la perte d'un élément : ligne de transmission, transformateur ou producteur. En plus du caractère non convexe hérité de l'OPF, le PSCOPF anticipe plusieurs cas « N-1 » probables, ce qui veut dire qu'il considère simultanément plusieurs états du réseau. Autrement dit, le problème PSCOPF consiste à résoudre simultanément plusieurs problèmes OPF couplés : les plans de production et de tension des cas « N-1 » dépendent des plans du cas normal. Ce problème a été donné dans la phase Beta de la compétition GO sous la forme suivante :

$$
\begin{aligned}
&\min \sum_{g\in G} c_g(Re(S_{g,0}^{gen})) \\
&S_{n,k}^{gen} = \mathbf{S_n^d} + \sum_{b\in B^-(n)} S_b^{dest}(v_k) + \sum_{b\in B^+(n)} S_b^{orig}(v_k) + S_n^{MCS}(v_k) && \forall n \in N, \forall k \in K \\
&\mathbf{v_n^{min}} \leq |v_{n,k}| \leq \mathbf{v_n^{max}} && \forall n \in N, \forall k \in K \\
&\mathbf{P_g^{min}} \leq Re(S_{g,k}^{gen}) \leq \mathbf{P_g^{max}} && \forall g \in G, \forall k \in K \\
&\mathbf{Q_g^{min}} \leq Im(S_{g,k}^{gen}) \leq \mathbf{Q_g^{max}} && \forall g \in G, \forall k \in K \\
&|S_b^{orig}(v_k)| \leq \mathbf{S_l^{max}} && \forall b \in B, \forall k \in K \\
&|S_b^{dest}(v_k)| \leq \mathbf{S_l^{max}} && \forall b \in B, \forall k \in K \\
&Re(S_{g,k}^{gen}) = Re(S_{g,0}^{gen}) + \boldsymbol{\alpha_g}\Delta_k && \forall g \in G, \forall k \in K^* \\
&|v_{g,k}| < |v_{g,0}| \Rightarrow Im(S_{g,k}^{gen}) = \mathbf{Q_g^{max}} && \forall g \in G, \forall k \in K^* \\
&|v_{g,k}| > |v_{g,0}| \Rightarrow Im(S_{g,k}^{gen}) = \mathbf{Q_g^{min}} && \forall g \in G, \forall k \in K^* \\
&v_k, S_k^{gen} \in \mathbb{C}^{|N|}, \Delta_k \in \mathbb{R} && \forall k \in K
\end{aligned}
\tag{PSCOPF}
$$

où les constantes sont en gras et $K = \{0, 1, 2, \ldots, N_K\}$ représente les différents états du réseau : l'index 0 représente le cas normal tandis que les autres indices représentent les cas

« N-1 ». La constante $\boldsymbol{\alpha_g}$ est le coefficient de participation du producteur g et la variable Δ_k représente la pénurie réelle d'électricité avant rétablissement dans le cas k. Les autres notations sont définies dans le chapitre 2 section 2. Il faut noter ici que les limites thermiques sont définies en puissance et non en courant ($|S_b^{orig}(V_k)|, |S_b^{dest}(V_k)| \leq \mathbf{S_l^{max}}$) et que les fonctions c_g sont quadratiques. Les deux dernières contraintes sont appelées « contraintes PV/PQ switching » : la production de puissance réactive s'ajuste de manière à maintenir le module de tension défini dans le cas normal. Autrement dit, une baisse du module de la tension est compensée par une injection maximale de puissance réactive et vice-versa. Nous modélisons ces contraintes avec des variables binaires.

3.4.2 Calcul de solutions réalisables

Nous avons construit les problèmes PSCOPF en variables complexes grâce à notre module Julia MathProgComplex.jl et à la flexibilité de modélisation décrite en section 3.3. Nous les avons convertis en problèmes avec variables réelles en utilisant la représentation rectangulaire, ce qui donne des Problèmes Non Linéaire en variables Mixtes (MINLP). Nous avons utilisé AMPL [85] et le solveur Knitro [91] pour la résolution et nous avons appliqué une méthode en 3 étapes pour calculer de bonnes solutions réalisables. En effet, puisque le problème PSCOPF combine les difficultés, l'idée est de commencer par résoudre un problème simple puis d'augmenter progressivement la difficulté. La première étape consiste à résoudre la relaxation continue du problème, ce qui revient à calculer des solutions pour tous les problèmes OPF comme s'ils n'étaient pas couplés. Cette solution est un bon point de départ pour la deuxième étape qui consiste à résoudre le problème avec variables binaires en utilisant l'option complémentarité du solveur Knitro. Cette option permet de convertir toutes les variables binaires en contraintes de complémentarité qui sont ajoutées à la fonction objectif avec un terme de pénalisation, transformant ainsi le problème en variables mixtes en problème continu. Le terme de pénalisation est mis à jour automatiquement par Knitro. La troisième étape consiste à fixer les variables binaires obtenues à la deuxième étape et à résoudre le problème continu obtenu. Cette étape permet de s'assurer que la solution finale est réalisable à la précision demandée. Pour résumer, la première étape permet de calculer un bon point de départ en relaxant les variables binaires et les deux étapes suivantes ont pour objectif d'obtenir la convergence avec les variables binaires.

3.4.3 Résultats numériques

Nous avons testé notre module sur les jeux de données fournis dans la phase Beta de la compétition GO. Plusieurs jeux de données étaient proposés pour des petits réseaux (avec

des solutions). Cette phase Beta s'est terminée à l'automne 2018 mais certains des jeux de données sont encore disponibles à l'adresse [92]. Le tableau 3.3 présente rapidement les 4 jeux de données qui ont été proposés. Ils représentent tous des petits réseaux : les jeux IEEE14 et Modified_IEEE14 représentent des réseaux à 14 nœuds et les jeux RTS96 et Modified_RTS96 représentent des réseaux à 73 nœuds. La principale différence entre un jeu et sa version modifiée est le nombre de lignes : la version modifiée représente le même réseau mais comporte une ligne en moins. Pour les jeux IEEE14 et Modified_IEEE14, il y a un seul cas « N-1 » à considérer (perte d'une ligne de transmission). Pour les jeux RTS96 et Modified_RTS96, il y a respectivement 10 et 9 cas « N-1 » : une ligne parmi 10 (ou 9) peut être perdue. Finalement, pour chaque jeu de données, il y a 100 scénarios qui correspondent à 100 instances différentes. Les différences peuvent être liées à la demande électrique par exemple.

Les tests ont été réalisés avec un processeur Intel® Core™ i7-6820HQ CPU @2.70GHz en utilisant notre module MathProgComplex.jl avec Julia 0.6.1, AMPL Version 20161231 et le solveur Knitro 10.3.0. La tolérance de réalisabilité (feastol) a été fixée à 10^{-6} (précision demandée dans la compétition) et la tolérance d'optimalité (optol) à 10^{-3}. On dit qu'une instance est résolue si l'on est capable de calculer une solution réalisable aussi bonne que celle fournie dans les jeux de données. Les résultats sont résumés dans le tableau 3.4 et ont été confirmés par la plateforme de test de la compétition GO. Des problèmes numériques ont été mis en évidence dans ces travaux. En effet, pour chaque jeu de données, la plupart des instances sont résolues mais pas toutes : 90/100 pour IEEE14 et RTS96, 89/100 pour Modified_RTS96 et seulement 84/100 pour Modified_IEEE14. La relaxation continue (étape 1) converge toujours mais la convergence est plus délicate à l'étape 2 avec les variables binaires. Plus précisément, elle dépend fortement du préconditionnement : en utilisant différents préconditionnements, le nombre d'instances résolues est différent et les instances qui ne sont pas résolues ne sont pas toujours les mêmes. Par exemple, pour IEEE14, seulement 78 instances sur 100 sont résolues sans préconditionnement.

Tableau 3.3 Description des jeux de données PSCOPF de la compétition GO

Réseau	#nœuds	#lignes	#cas « N-1 »	#scénarios
IEEE14	14	20	1	100
Modified_IEEE14	14	19	1	100
RTS96	73	120	10	100
Modified_RTS96	73	119	9	100

Tableau 3.4 Résultats obtenus sur les jeux de données PSCOPF de la compétition GO

Réseau	#var.	#ctr.	#coeff. non nuls Jacobienne	#coeff. non nuls Hessienne	#scénarios résolus
IEEE14	92	207	937	245	90/100
Modified_IEEE14	92	203	905	237	84/100
RTS96	4784	12157	49838	7199	90/100
Modified_RTS96	4340	10987	44960	6512	89/100

3.4.4 Impact du préconditionnement sur la résolution

Nous avons comparé trois préconditionnements différents : aucun préconditionnement, un préconditionnement sur les variables pour toutes les étapes (préconditionnement 1) et un préconditionnement sur les variables et les contraintes à partir de l'étape 2 (préconditionnement 2). Plus précisément, le préconditionnement 1 consiste à diviser chaque variable $x[i]$ par la différence entre sa borne supérieure et sa borne inférieure :

$$x_{scaled}[i] = \frac{x[i]}{ub[i] - lb[i]}. \tag{3.1}$$

Le préconditionnement 2 utilise les valeurs calculées à l'étape 1 (variables primales $\overline{x}_1$ et variables duales $\overline{\lambda}_1$) pour redimensionner les variables x et les contraintes c :

$$\begin{aligned} x_{scaled}[i] &= \tfrac{x[i]}{|\overline{x}_1[i]|} \text{ si } |\overline{x}_1[i]| \geq 10^{-5} \text{ sinon } x[i] \\ c_{scaled}[i] &= \tfrac{c[i]}{|\overline{\lambda}_1[i]|} \text{ si } |\overline{\lambda}_1[i]| \geq 10^{-5} \text{ sinon } c[i]. \end{aligned} \tag{3.2}$$

La figure 3.1 présente l'impact du préconditionnement sur le nombre d'itérations de l'étape 2 pour le jeu de données IEEE14 (100 scénarios). Le nombre d'itérations est rangé par ordre décroissant pour chaque type de préconditionnement (aucun préconditionnement, préconditionnement 1, préconditionnement 2). Le nombre maximum d'itérations autorisées dans l'étape 2 est 600, ce qui veut dire que les scénarios pour lesquels 600 itérations ont été réalisées sont des scénarios qui n'ont pas été résolus. Ainsi, nous pouvons observer qu'il y a beaucoup plus de scénarios non résolus sans préconditionnement. De manière générale, le nombre d'itérations est plus élevé sans préconditionnement. Les deux préconditionnements améliorent les performances mais le préconditionnement 1 semble plus efficace, surtout au niveau du nombre d'instances qui ne sont pas résolues en 600 itérations. Cependant, aucun des deux préconditionnements ne permet de résoudre toutes les instances en même temps.

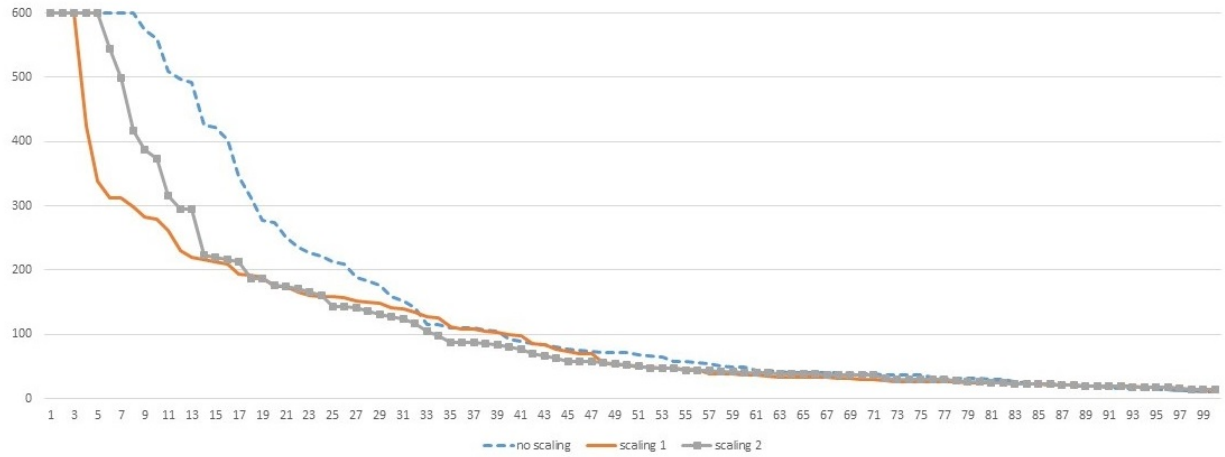

Figure 3.1 Impact du préconditionnement sur le nombre d'itérations de l'étape 2 pour le jeu de données IEEE14 : Nombre d'itérations pour l'étape 2 trié par ordre décroissant pour les 100 scénarios IEEE14 pour trois conditionnements différents

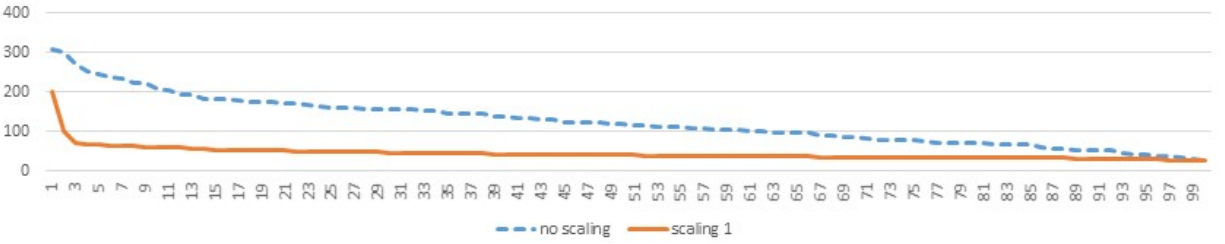

Figure 3.2 Impact du préconditionnement sur le nombre d'itérations de l'étape 1 pour le jeu de données RTS96 : Nombre d'itérations pour l'étape 1 trié par ordre décroissant pour les 100 scénarios RTS96 pour deux conditionnements différents

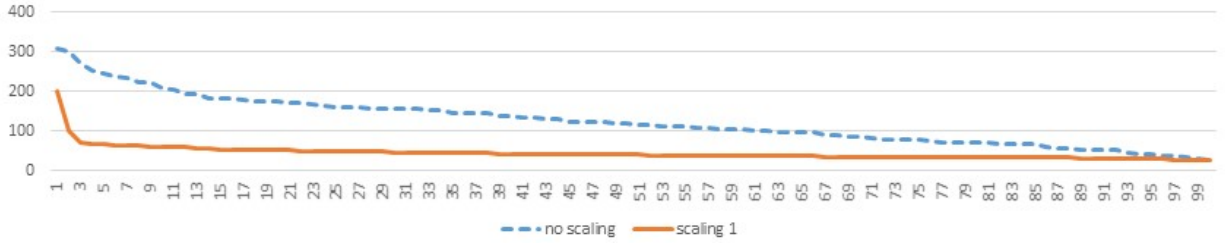

Figure 3.3 Impact du préconditionnement sur le temps CPU (en secondes) de l'étape 1 pour le jeu de données RTS96 : Temps CPU pour l'étape 1 trié par ordre décroissant pour les 100 scénarios RTS96 pour deux conditionnements différents

Le préconditionnement peut aussi avoir un effet positif sur la résolution de l'étape 1 : même si cette étape converge pour toutes les instances, l'utilisation d'un préconditionnement adapté peut accélérer la résolution en terme de nombre d'itérations et de temps CPU. Les figures 3.2 et 3.3 présentent l'impact du préconditionnement sur l'étape 1 pour les 100 scénarios du jeu de données RTS96. Elles montrent respectivement le nombre d'itérations et le temps CPU triés par ordre décroissant sans préconditionnement et avec le préconditionnement 1. La figure 3.2 démontre que le préconditionnement 1 permet de réduire significativement le nombre d'itérations nécessaires pour converger. De manière similaire, la figure 3.3 montre une diminution du temps CPU avec le préconditionnement 1 (ce qui est cohérent avec la diminution du nombre d'itérations). Ainsi, le préconditionnement a un impact significatif sur l'étape 2 qui est l'étape problématique mais aussi sur l'étape 1. Cependant, il est difficile de trouver un préconditionnement générique qui soit performant sur toutes les instances. Nous avons testé d'autres préconditionnements que ceux présentés ici mais aucun n'a permis de résoudre toutes les instances. Le plus efficace que nous ayons trouvé est le préconditionnement 1. C'est celui qui a été utilisé pour trouver les résultats du tableau 3.4.

3.5 Conclusion et perspectives

Notre module MathProgComplex.jl et notre structure générique pour les OPF nous ont permis d'implémenter simplement et rapidement un problème d'OPF assez compliqué, le PSCOPF. Nous avons testé et validé les problèmes obtenus sur les instances de la phase Beta de la compétition GO avec la plateforme de test associée. Dans la plupart des cas nous pouvons calculer des solutions réalisables aussi bonnes que celles données en solution dans la compétition en utilisant une méthode simple. Ces travaux ont été publiés à la conférence PowerTech 2019 [93]. Dans le chapitre 5, nous nous intéressons à un autre problème d'OPF qui peut également être implémenté simplement grâce à notre module et à notre structure générique pour les OPF.

Même si notre but n'était pas de résoudre le problème PSCOPF mais seulement de valider nos modules, il est possible d'aller plus loin dans la résolution. D'abord, il serait intéressant de poursuivre les travaux sur le conditionnement pour trouver un préconditionnement efficace sur toutes les instances. Ensuite, résoudre des relaxations convexes permettrait d'obtenir des bornes inférieures et ainsi d'être en mesure d'évaluer la qualité des solutions réalisables obtenues. Pour clore un éventuel écart entre la borne supérieure et la borne inférieure, une méthode d'optimisation globale de type énumération implicite pourrait être envisagée. Finalement, il serait judicieux de mettre en place des méthodes de décomposition afin de pouvoir traiter des réseaux de grande taille et/ou des problèmes avec beaucoup de cas N-1. Nous

reviendrons sur l'importance de la décomposition dans le chapitre suivant à propos des relaxations SDP de la hiérarchie de Lasserre.

Par ailleurs, d'autres fonctions pourraient être ajoutées à notre module MathProgComplex.jl : par exemple la conversion d'un problème complexe en utilisant la représentation polaire pour toutes les variables ou pour certaines seulement. Notre structure générique pour les OPF pourrait aussi être complétée par un certain nombre de relaxations convexes qui pourraient être générées automatiquement à partir d'un problème d'OPF.

CHAPITRE 4 RÉSOLUTION DES RELAXATIONS SDP DE L'OPF

Dans ce chapitre, nous nous intéressons principalement à la résolution de la relaxation du rang pour le problème d'OPF décrit dans la section 2.2. Après avoir exposé nos motivations, nous proposons un bref état de l'art des solveurs SDP compatibles avec le module d'optimisation JuMP. Nous montrons ensuite que le choix de la décomposition en cliques a un impact significatif sur le temps de résolution et nous proposons une nouvelle stratégie pour améliorer les performances. Pour finir, nous discutons d'une possible extension de nos travaux à la résolution de relaxations SDP d'ordre 2 ou plus de la hiérarchie de Lasserre.

4.1 Motivations

De manière générale, la relaxation du rang est souvent délaissée au profit de relaxations convexes moins précises mais plus rapides à résoudre. Pour les problèmes d'OPF, la relaxation du rang est particulièrement précise, voire exacte mais de nombreux travaux de recherche ont proposé des relaxations moins coûteuses [35, 37, 38] afin de traiter des réseaux toujours plus grands. Voir la récente étude [28] pour une liste exhaustive des relaxations proposées. Cependant, peu de travaux ont cherché à accélérer la résolution de la relaxation du rang pour l'OPF en utilisant des algorithmes de points intérieurs. A notre connaissance, les seuls travaux dans cette direction sont [1] et [94]. Ce dernier propose une reformulation du problème d'OPF plus adaptée aux méthode de points intérieurs implémentées dans les meilleurs solveurs SDP. La méthode proposée dans [1] est très différente : une stratégie de fusion de cliques est appliquée pour réduire le nombre de contraintes liantes d'une décomposition en cliques maximales. Cette stratégie semble efficace mais l'article date de 2013 et la plus grande instance testée comporte seulement 300 nœuds. Nous proposons de continuer à explorer cette piste de recherche afin de tirer parti de la précision de la relaxation du rang, que cela soit à l'intérieur d'une méthode d'optimisation globale ou bien simplement pour calculer une borne inférieure. Notre objectif principal est donc d'accélérer la résolution de la relaxation du rang, en particulier pour les réseaux de grande taille. Un deuxième objectif, plus ambitieux, serait d'accélérer la résolution des relaxations SDP d'ordre 2 ou plus de la hiérarchie de Lasserre.

4.2 Etat de l'art des outils pour résoudre la relaxation du rang de l'OPF

4.2.1 Construction des relaxations SDP de l'OPF

Notre module MathProgComplex.jl et le module JuMP.jl permettent de construire les relaxations SDP de l'OPF (relaxations du rang). D'abord, nous utilisons notre module pour construire les problèmes d'OPF en variables complexes à partir du format d'entrée MATPOWER [21]. Nous les convertissons ensuite en problèmes avec des variables réelles puis nous construisons leurs relaxations SDP sous forme de modèles JuMP en spécifiant la décomposition en cliques à utiliser. Finalement, nous utilisons un solveur SDP compatible avec JuMP pour la résolution. Plusieurs solveurs SDP sont compatibles avec le module JuMP. Nous avons donc décidé de les comparer. Tous les tests ont été réalisés avec un processeur Intel® Core™ i7-6820HQ CPU @2.70GHz.

4.2.2 Méthode de référence pour la décomposition en cliques maximales

La méthode que nous présentons ici est la méthode la plus communément utilisée pour calculer des décompositions en cliques pour les relaxations SDP de l'OPF [1, 95]. Elle sert de référence pour toutes nos comparaisons.

Pour commencer, la décomposition en cliques est calculée sur la relaxation SDP de l'OPF formulé en variables complexes. Cette décomposition est ensuite convertie en variables réelles en doublant la taille de chaque clique pour y faire apparaître la partie réelle et la partie imaginaire de chaque variable. Lorsque l'on travaille en variables complexes, le graphe G^A correspond au graphe du réseau de transport d'électricité. L'heuristique d'extension cordale utilisée s'appuie sur une factorisation de Cholesky [96, 97] et un ordre AMD (Approximation de l'ordre Degré Minimal) [66] est utilisé pour permuter les lignes et les colonnes avant la factorisation. L'algorithme de Prim est utilisé pour calculer un arbre de cliques.

Nous avons implémenté cette procédure en utilisant Julia 1.0.3. et les modules SuiteSparse.jl, LightGraphs.jl [98] et MetaGraphs.jl.

4.2.3 Limites des solveurs SDP disponibles pour JuMP v0.19.0

Nous utilisons Julia v1.0.3 et JuMP.jl v0.19.0 qui sont les versions les plus stables [novembre 2019]. Les solveurs disponibles sont listés dans le tableau 4.1 avec l'algorithme utilisé et leur langage d'origine.

Nous avons comparé les solveurs suivants sur plusieurs relaxations SDP de l'OPF avec Julia v1.0.3 et JuMP.jl v0.19.0 :

Tableau 4.1 Solveurs SDP disponibles avec JuMP v.0.19.0 [novembre 2019]

Solveur	Algorithme	Langage d'origine
CSDP [99]	Méthode barrière primale-duale prédicteur-correcteur	C
MOSEK [100]	Méthode de points intérieurs primale-duale	C/C++
SCS [101]	ADMM	C
SeDuMi [102]	Méthode de points intérieurs	MATLAB

— CSDP.jl v0.4.1 ;
— Mosek.jl v0.9.11 avec MOSEK 8.1.0.72 ;
— SCS.jl v0.5.1 ;
— SeDuMi.jl v0.1.0.

Nous avons utilisé les paramètres par défaut et nous avons testé les petites instances MATPOWER case9, case30 et case57 qui représentent respectivement des réseaux électriques de 9, 30 et 57 nœuds. Pour chaque instance, nous avons testé la résolution avec et sans décomposition en cliques. La décomposition en cliques utilisée est celle décrite dans la sous-section précédente. Tous les tests ont été réalisés avec un processeur Intel® Core™ i7-6820HQ CPU @2.70GHz. Les résultats sur l'instance case57 sont présentés dans le tableau 4.2 et ils montrent que le solveur MOSEK est le seul solveur qui converge avec et sans décomposition et c'est aussi le solveur le plus rapide. Les résultats sur les autres instances confirment que le solveur MOSEK est le plus fiable et le plus performant. Il est possible que les autres solveurs soient plus performants dans leur langage d'origine mais il est préférable pour nous d'utiliser un seul langage donc nous utilisons uniquement le solveur MOSEK pour toutes nos résolutions de problèmes SDP.

Nous avons aussi voulu observer les limites actuelles [novembre 2019] du solveur MOSEK sur les problèmes sans décomposition. L'heure de résolution est dépassée pour les instances de plus de 1500 nœuds. La seule construction des problèmes requiert du temps et de la mémoire, puis plusieurs heures sont nécessaires pour la résolution. Par exemple, les temps de résolution pour les instances case2868rte et case3012wp sont présentés dans le tableau 4.3. Les premières colonnes décrivent la taille de l'instance : nombre de variables scalaires (#var) et nombre de contraintes (#ctr). Les deux dernières colonnes présentent le temps de résolution donné par la fonction JuMP 'optimize' et le temps total (construction + résolution). Ces deux temps sont donnés en heure. Aucun problème de convergence ni de mémoire n'ont été observés pour ces deux instances (avec 32Go de mémoire RAM). Le temps apparaît donc ici comme le facteur limitant. Toutefois, il est probable que des problèmes de mémoire apparaissent pour

Tableau 4.2 Comparaison de plusieurs solveurs SDP sur l'instance case57 (avec et sans décomposition)

Sans décomposition				
Solveur	Objectif	Statut	Temps (s)	Mémoire (Mo)
CSDP	25337.79	OPTIMAL	9.11	23.1
Mosek	25337.79	OPTIMAL	0.334	5.68
SCS	25334.87	ALMOST_OPTIMAL	117.9	18.7
SeDuMi	25281.20	ALMOST_OPTIMAL	757.3	29.8
Avec décomposition				
Solveur	Objectif	Statut	Temps (s)	Mémoire (Mo)
CSDP	945806.97	SLOW_PROGRESS	141.3	139.8
Mosek	25337.79	OPTIMAL	0.284	12.5
SCS	25337.94	OPTIMAL	10.14	6.58
SeDuMi	24965.72	ALMOST_OPTIMAL	0.939	24.6

des instances de plus de 6000 nœuds.

Pour comparaison, ces instances sont résolues en quelques minutes avec la décomposition en cliques décrite dans la sous-section 4.2.2.

4.2.4 Briser la symétrie de rotation : choix d'un nœud de référence

Dans la plupart des problèmes d'OPF, une symétrie de rotation est observée. En effet, la modélisation ne fait intervenir que des produits hermitiens $v_i\overline{v}_j$. Ainsi si v est solution alors $ve^{i\alpha}$ est aussi solution quel que soit le réel α car $v_i e^{i\alpha}\overline{v_j e^{i\alpha}} = v_i e^{i\alpha}\overline{v}_j e^{-i\alpha} = v_i\overline{v}_j$. Les symétries peuvent avoir un impact négatif sur la résolution et il est donc courant d'essayer de les briser. Pour l'OPF, il est courant de briser la symétrie de rotation en choisissant un nœud de référence, c'est-à-dire en fixant la partie imaginaire de ce nœud à 0. La question est alors de savoir quel nœud choisir. Choisir le nœud qui a le plus grand degré dans le graphe du réseau est une option raisonnable et très répandue. Dans le contexte de la décomposition en cliques, une alternative tout aussi raisonnable est de choisir le nœud qui a le plus de voisins dans

Tableau 4.3 Résolution des instances case2868rte et case3012wp avec MOSEK sans décomposition

Instance	#var	#ctr	Temps de résolution (h)	Temps total (h)
case2868rte	19620	16859	7.67	21.27
case3012wp	20288	19672	17.36	46.86

l'extension cordale du graphe du réseau. Nous avons testé ces deux options mais nous n'avons observé aucune amélioration significative. Au contraire, le premier choix avait plutôt tendance à détériorer les temps de résolution. Ceci n'est pas forcément étonnant : certains algorithmes sont fortement ralentis par les symétries (e.g. les algorithmes de Branch-and-Bound) mais il est possible que les symétries soient plutôt bénéfiques pour les méthodes de points intérieurs car elles pourraient permettre de trouver une solution réalisable plus rapidement. Ainsi, dans toute la suite de ce document, nous abandonnons l'idée de briser la symétrie de rotation et nous ne choisissons pas de nœud de référence.

4.3 Impact de la décomposition en cliques maximales sur la résolution de la relaxation du rang de l'OPF

Dans cette section, nous testons différentes décompositions en cliques maximales pour résoudre la relaxation du rang de l'OPF et nous montrons l'impact du choix de cette décomposition sur la résolution des problèmes SDP.

4.3.1 Application de la méthode de référence : analyse de la structure des relaxations SDP de l'OPF

Les réseaux de transport d'électricité sont maillés (i.e. ils contiennent des cycles) car ils doivent respecter des contraintes de « N-1 » : de manière grossière, si une ligne n'est plus disponible, le courant doit pouvoir passer ailleurs. Toutefois, ces réseaux sont peu maillés car les lignes de transport d'électricité sont des ouvrages onéreux. Les problèmes d'optimisation définis sur ces réseaux, y compris les problèmes d'OPF, sont donc très creux. La relaxation du rang des problèmes d'OPF hérite de ce caractère creux, tout comme les décompositions en cliques maximales calculées avec la méthode de référence (définie dans la section 4.2.2.). En effet, celles-ci contiennent beaucoup de petites cliques. Le tableau 4.4 résume pour quelques instances MATPOWER 6.0 [21] les caractéristiques de ces décompositions. Nous ne présentons que 5 instances car les tendances sont les mêmes sur toutes les instances de la librairie MATPOWER 6.0. La première colonne donne le nom de l'instance. Le nombre qui apparaît dans chaque nom d'instance est le nombre de nœuds du réseau considéré. La deuxième (resp. troisième) colonne donne le nombre de variables scalaires (resp. contraintes) du problème SDP. La quatrième colonne présente le nombre de cliques obtenu pour chaque instance avec la méthode de référence. Ce nombre est le même que l'on travaille en nombres complexes ou en nombres réels. La cinquième (resp. sixième) colonne donne la taille de la plus grosse (resp. petite) clique en variables complexes. Pour obtenir ces tailles en variables réelles, il suffit de doubler les tailles en variables complexes. Finalement, la dernière colonne représente

le pourcentage de cliques qui ont une taille inférieure ou égale à 5 en variables complexes. Ce tableau montre que le nombre de cliques est du même ordre de grandeur que le nombre de nœuds dans le réseau. Les plus grosses cliques ont des tailles raisonnables : 35 en variables complexes ou 70 en variables réelles pour les deux plus gros réseaux. Enfin, les décompositions contiennent beaucoup de petites cliques : pour chaque instance, environ 95% des cliques sont de taille inférieure ou égale à 5 en complexes (soit des cliques de taille inférieure ou égale à 10 en réels).

4.3.2 Comparaison de différentes heuristiques pour l'extension cordale

Il existe plusieurs heuristiques pour calculer des extensions cordales et la décomposition en cliques peut être très différente en fonction de l'heuristique utilisée. Bien que les décompositions soient équivalentes en théorie, elles ne le sont pas en terme de résolution. Nous le montrons ici en comparant deux heuristiques : l'heuristique qui s'appuie sur la factorisation de Cholesky et l'ordre AMD (notre méthode de référence) et l'heuristique Degré Minimal (MD) [103] décrite dans l'algorithme 1. Nous avons choisi ces deux heuristiques car elles font partie des meilleures heuristiques connues pour calculer une extension cordale avec peu d'arêtes additionnelles. Pour ces deux heuristiques, l'extension cordale obtenue dépend de l'ordre dans lequel les sommets du graphe initial sont visités. Le tableau 4.5 présente des résultats de cette comparaison sur des instances MATPOWER de plus de 1000 nœuds. La version 8.1 de MOSEK a été utilisée. Soit nc^{MD}, nlc^{MD} et t^{MD} le nombre de cliques maximales, le nombre de contraintes liantes et le temps de résolution MOSEK respectivement obtenus avec l'heuristique MD. De manière similaire soit nc^{AMD}, nlc^{AMD} et t^{AMD} les mêmes quantités mais obtenues avec la factorisation de Cholesky et AMD. Le tableau 4.5 présente des ratios entre ces trois quantités. Tous les ratios sont exprimés en pourcentages. Ce tableau n'est pas exhaustif mais il montre que les décompositions en cliques peuvent être différentes

Tableau 4.4 Caractéristiques des décompositions en cliques obtenues avec la méthode de référence sur des instances MATPOWER

Instance	Nb de variables	Nb de contraintes	Nb de cliques	Taille maximale	Taille minimale	Cliques taille ≤ 5
case1888rte	38072	31643	1816	13	2	97%
case3012wp	98155	77882	2916	28	2	95%
case6468rte	172849	128242	6153	30	2	96%
case9241pegase	342607	259363	8577	35	2	93%
case13659pegase	380676	281128	12997	35	2	95%

d'une heuristique à l'autre : le nombre de cliques et le nombre de contraintes sont différents pour ces deux décompositions. Même si elles ont à peu près le même nombre de cliques, il y a des différences significatives pour le temps de résolution MOSEK. Ces différences sont en partie expliquées par la différence dans le nombre de contraintes liantes, surtout pour les plus grosses instances. La taille de la plus grosse clique peut être un autre facteur.

Entrées : Un graphe G=(V,E)
Sorties : Une extension cordale H

$G_\alpha \leftarrow G$
$H \leftarrow G$
pour $k = 1 : |V|$ **faire**
  $v \leftarrow$ sommet de degré minimal dans G_α
  $F \leftarrow$ ensemble des arêtes à ajouter dans G_α pour que tous les voisins de v soient reliés entre eux
  $G_\alpha \leftarrow G_\alpha + F$
  $G_\alpha \leftarrow G_\alpha - v$
  $H \leftarrow H + F$
fin

Algorithme 1 : Heuristique Degré Minimal

Il est possible de générer des décompositions qui diffèrent beaucoup plus en terme de nombre de cliques et de nombre de contraintes liantes. Dans ce cas, les différences de temps de résolution sont encore plus significatives. Dans des tests préliminaires nous avons implémenté des "mauvaises" décompositions en essayant plusieurs ordres avant la factorisation de Cholesky (ordre aléatoire, ordre maximal des degrés par exemple). Ici une "mauvaise" décomposition est une décomposition avec beaucoup de contraintes liantes, ce qui peut causer des problèmes de mémoire avec le solveur MOSEK.

Tableau 4.5 Comparaison de décompositions en cliques pour l'heuristique MD et l'heuristique de référence AMD

Instance	$\frac{nc^{MD}}{nc^{AMD}} - 1$	$\frac{nlc^{MD}}{nlc^{AMD}} - 1$	$\frac{t^{MD}}{t^{AMD}} - 1$
case1888rte	0.17%	1.7%	6.1%
case3012wp	-0.14%	-3.3%	-15.1%
case6468rte	0.10%	0.31%	-6.8%
case9241pegase	-0.02%	4.9%	62%
case13659pegase	-0.04%	4.7%	48%

4.3.3 Comparaison des décompositions en cliques obtenues à partir de l'OPF formulé en variables complexes à celles obtenues à partir de l'OPF formulé en variables réelles

Les relaxations SDP de l'OPF sont naturellement définies en variables complexes mais elles doivent être exprimées en variables réelles pour être résolues par les solveurs SDP. Lorsque l'on réécrit le modèle avec des variables réelles, la matrice de parcimonie A change. De manière évidente, elle est deux fois plus grosse pour le problème SDP en variables réelles. Cette matrice peut amener des décompositions qui ne seraient pas possibles avec la formulation originale en variables complexes : par exemple, la partie réelle et la partie imaginaire d'un même nœud pourraient apparaître dans deux cliques différentes. Nous avons donc deux possibilités pour calculer la décomposition en cliques sur un problème SDP en variables complexes : soit nous appliquons la procédure sur la formulation SDP complexe et nous convertissons les cliques complexes obtenues en cliques réelles, soit nous appliquons directement la procédure sur la formulation réelle. Ces deux alternatives ne sont pas équivalentes. Nous le montrons théoriquement sur un petit exemple MATPOWER [21] (LMBM3) pour commencer.

Soit G^c (respectivement G^r) le graphe associé à la matrice de parcimonie du problème SDP complexe (respectivement réel). Soit H^c (respectivement H^r) l'extension cordale calculée à partir de G^c (respectivement G^r) avec un algorithme donné. L'extension cordale H^c peut être convertie en nombres réels de la même façon que le graphe complexe G^c peut être converti en G^r. Soit H^c_{real} cette conversion de H^c en réels. Le graphe H^c_{real} est aussi une extension cordale de G^r. L'objectif ici est de comparer H^c_{real} et H^r.

Pour l'instance LMBM3, G^c est une clique de taille 3, notée K_3. Pour cette même instance, G^r est le graphe représenté dans la Figure 4.1 avec 6 sommets : $V^r = \{1_{Re}, 1_{Im}, 2_{Re}, 2_{Im}, 3_{Re}, 3_{Im}\}$ et 12 arêtes :

$$\begin{aligned} E^r = \; & \{(1_{Re}, 2_{Re}), (1_{Re}, 2_{Im}), (1_{Im}, 2_{Re}), (1_{Im}, 2_{Im}), \\ & (1_{Re}, 3_{Re}), (1_{Re}, 3_{Im}), (1_{Im}, 3_{Re}), (1_{Im}, 3_{Im}), \\ & (2_{Re}, 3_{Re}), (2_{Re}, 3_{Im}), (2_{Im}, 3_{Re}), (2_{Im}, 3_{Im})\}. \end{aligned}$$

Puisque G^c est complet et donc cordal, $H^c = G^c = K_3$ peu importe l'algorithme d'extension cordale utilisé. Il y a donc une clique maximale avec 3 nœuds et aucune contrainte liante. Lorsque l'on convertit cette décomposition en variables réelles, on obtient $H^c_{real} = K_6$, la clique de taille 6, ce qui revient à ajouter 3 arêtes dans G^r. Cette décomposition donne une clique maximale avec 6 variables réelles et aucune contrainte liante.

D'autre part, G^r n'est pas cordal car il y a plusieurs cycles induits de taille 4, e.g. $1_{Re} - 2_{Im} - 1_{Im} - 2_{Re}$. Il suffit d'ajouter les deux arêtes $(2_{Re}, 2_{Im})$ et $(3_{Re}, 3_{Im})$ à G^r pour obtenir une

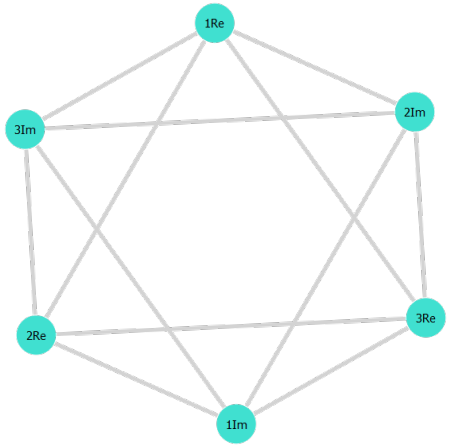

Figure 4.1 Le graphe G^r pour LMBM3 - Graphe obtenu avec le module GraphPlot.jl

extension cordale minimale H^r. Nous pouvons démontrer que H^r est cordal en utilisant la caractérisation suivante : un graphe est cordal si et seulement si il possède un ordonnancement d'élimination parfaite. Un ordonnancement d'élimination parfaite est un ordonnancement des sommets d'un graphe tel que, quel que soit le sommet v, les voisins de v qui se trouvent après lui dans l'ordonnancement forment une clique. Or $1_{Re} - 1_{Im} - 2_{Re} - 2_{Im} - 3_{Re} - 3_{Im}$ est un ordonnancement d'élimination parfaite pour le graphe H^r. Cette extension cordale donne deux cliques maximales : $\{2_{Re}, 2_{Im}, 3_{Re}, 3_{Im}, 1_{Re}\}$ et $\{2_{Re}, 2_{Im}, 3_{Re}, 3_{Im}, 1_{Im}\}$ et 10 contraintes liantes. Il y a d'autres possibilités pour obtenir une extension cordale minimale H^r, e.g., si les arêtes $(1_{Re}, 2_{1m})$ et $(3_{Re}, 3_{Im})$ sont ajoutées, mais les conclusions sont les mêmes.

Ce petit exemple prouve que ce n'est pas équivalent de calculer une décomposition en cliques sur le problème complexe ou sur le problème réel du point de vue théorique : on peut ajouter moins d'arêtes dans le cas réel pour obtenir une extension cordale. Cette idée est confirmée par des tests numériques qui montrent qu'il y a en moyenne 50% plus d'arêtes ajoutées dans l'extension cordale quand elle est faite sur G^c plutôt que sur G^r pour les instances MATPOWER 6.0 de plus de 1000 nœuds.

Cependant, ajouter moins d'arêtes dans l'extension cordale n'est pas nécessairement gagnant du point de vue résolution numérique. En effet, ajouter moins d'arêtes peut amener à des décompositions avec plus de cliques et plus de contraintes liantes comme le montre le tableau 4.6. Ce tableau présente des comparaisons entre des décompositions en cliques obtenues à partir du problème complexe (calculées à partir de G^c) et des décompositions obtenues à partir du problème réel (calculées à partir de G^r). Ces deux décompositions sont calculées avec notre méthode de référence (ordre AMD + factorisation de Cholesky). Soit L^c (respectivement L^r)

la liste des cliques maximales obtenue à partir de l'extension cordale H^c (respectivement H^r). La deuxième colonne représente le ratio pour le nombre de cliques défini comme $\frac{|L^r|}{|L^c|} - 1$. De manière similaire, la troisième colonne représente le ratio pour le nombre de contraintes liantes, i.e., $\frac{nlc^r}{nlc^c} - 1$ avec nlc^c (respectivement nlc^r) le nombre de contraintes liantes quand la décomposition est calculée pour le problème complexe (respectivement réel). Finalement, la quatrième colonne représente le ratio pour le temps de résolution MOSEK, i.e., $\frac{t^r}{t^c} - 1$ avec t^c (respectivement t^r) le temps de résolution pour la décomposition calculée pour le problème complexe (respectivement réel). Tous les résultats sont présentés en pourcentage.

Tableau 4.6 Comparaison de décompositions en cliques issues de G^c ou de G^r

Instance	$\frac{\lvert L^r\rvert}{\lvert L^c\rvert} - 1$	$\frac{nlc^r}{nlc^c} - 1$	$\frac{t^r}{t^c} - 1$
case1888rte	51%	24%	17%
case3012wp	39%	15%	4%
case6468rte	40%	19%	58%
case9241pegase	48%	18%	28%
case13659pegase	59%	20%	23%

Ce tableau montre que la décomposition en cliques sur G^r donne environ 50% plus de cliques et 20% plus de contraintes liantes que sur G^c, ce qui ralentit la résolution (le temps de résolution augmente d'au moins 4% et jusqu'à 58%). Ainsi, même si moins d'arêtes sont ajoutées dans le graphe G^r lorsque l'on travaille directement en variables réelles, il semble préférable de travailler en variables complexes pour avoir des meilleures performances. Cette comparaison démontre qu'il n'est pas nécessairement profitable de calculer une extension cordale avec un nombre minimal d'arêtes ajoutées.

4.3.4 Fusion de cliques

Les sections précédentes montrent que les performances de résolution sont très sensibles à la décomposition en cliques. Plus particulièrement, la section précédente montre qu'il peut être avantageux de calculer des extensions cordales avec plus d'arêtes. Pour explorer cette direction, nous nous intéressons aux procédures de fusion de cliques qui permettent de contrôler le nombre de cliques et de contraintes liantes a posteriori. En effet, l'objectif de ces procédures est d'améliorer une décomposition en cliques donnée en fusionnant des cliques, c'est-à-dire en ajoutant des arêtes dans l'extension cordale calculée. Par exemple, la figure 4.2 illustre la fusion de deux cliques de taille 3 ($\{5, 6, 9\}$ et $\{6, 7, 9\}$) en une clique de taille 4 ($\{5, 6, 7, 9\}$),

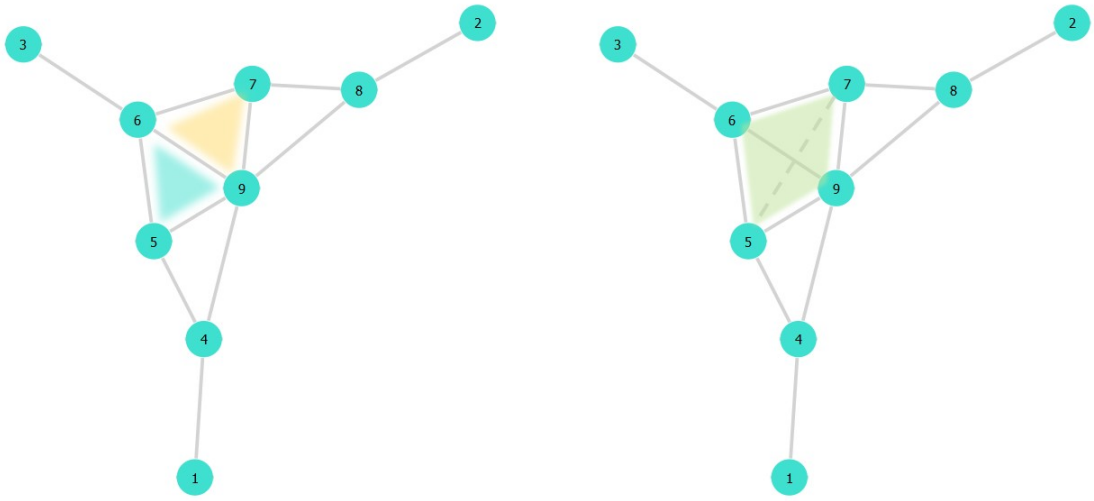

Figure 4.2 Exemple de fusion de deux cliques de taille 3 en une clique de taille 4

ce qui revient à ajouter une arête dans le graphe. Des heuristiques de regroupement de cliques sont proposées dans [1, 16].

Toutefois, il n'est pas possible de fusionner n'importe quelles cliques. Le contre-exemple ci-dessous (figures 4.3, 4.4, 4.5, 4.6) montre qu'il est possible de perdre la cordalité du graphe si on fusionne des cliques sans faire attention.

Le graphe G défini dans la figure 4.3 est cordal car l'ordre 6-7-3-4-1-2-5 est un ordonnancement d'élimination parfaite :

- les voisins de 6 dans le graphe G sont 3 et 5 qui sont adjacents
- les voisins de 7 dans G-{6} sont 4 et 5 qui sont adjacents
- les voisins de 3 dans G-{6,7} sont 1 et 5 qui sont adjacents
- les voisins de 4 dans G-{6,7,3} sont 2 et 5 qui sont adjacents
- les voisins de 1 dans G-{6,7,3,4} sont 2 et 5 qui sont adjacents
- 2 a un seul voisin dans G-{6,7,3,4,1}

Or le graphe n'est plus cordal après fusion des cliques {3,5,6} et {4,5,7} (voir figures 4.4 et 4.5) car il y a un cycle induit de 4 sommets : 1-2-4-3 (en rouge dans la figure 4.6).

Il existe cependant des conditions dans lesquelles la fusion de cliques est possible. Les auteurs de [16] ont prouvé que la fusion de cliques adjacentes dans l'arbre de cliques était valide et ont proposé une heuristique de fusion de cliques pour les problèmes SDP génériques. Des

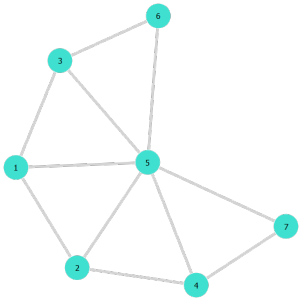

Figure 4.3 Graphe cordal G

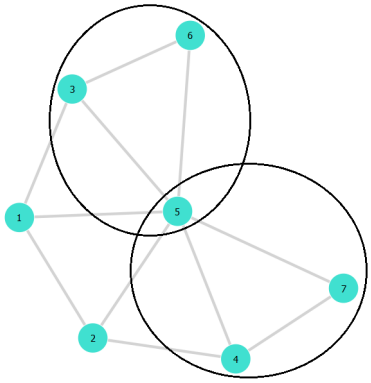

Figure 4.4 Les 2 cliques maximales qu'on va fusionner

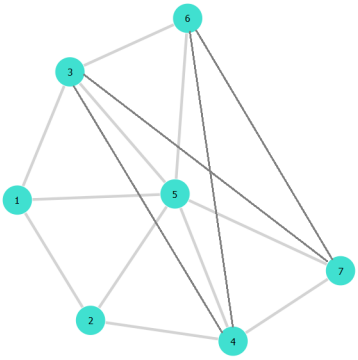

Figure 4.5 Graphe après fusion des 2 cliques

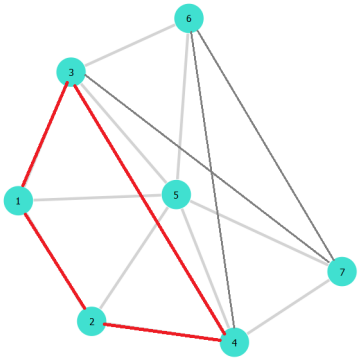

Figure 4.6 Cycle induit de 4 sommets sans corde

heuristique de fusion de cliques dans le même esprit ont été proposées dans [104, 105]. Une heuristique spécifique aux problèmes d'OPF est proposée dans [1] et selon les auteurs, elle est meilleure que les procédures génériques comme celle proposée dans [16, 104, 105]. Dans la suite, nous présentons donc l'heuristique proposée dans [1] puis nous présentons une autre approche de fusion de cliques que nous avons développée pour obtenir plus de diversité dans les décompositions obtenues.

Heuristique de fusion de cliques proposée par Molzahn dans [1] L'idée de cette heuristique est de fusionner les deux cliques qui minimisent un certain critère f à chaque itération. L'algorithme s'arrête lorsque le nombre de cliques souhaité est atteint (il est conseillé de prendre 10% du nombre de nœuds dans le réseau). Ce critère f est une estimation du coût de résolution d'un problème SDP. Il prend en compte le nombre de variables scalaires (les coefficients dans les matrices SDP) et le nombre de contraintes liantes de la façon suivante :

$$f_M(L, \ell) = m + \ell + \sum_{C_i \in L} |C_i|(|C_i| + 1)/2 \tag{4.1}$$

où ℓ dépend de l'arbre de cliques T :

$$\ell = \sum_{(C_i, C_j) \in E} d_{ij}(2d_{ij} + 1) \tag{4.2}$$

avec d_{ij} le nombre de variables complexes partagées par C_i et C_j.

Cette heuristique permet de réduire de près de 2300 secondes le temps total de résolution pour les instances MATPOWER 6.0.

Autres heuristiques de fusion de cliques Nous avons testé une autre approche de fusion de cliques dont l'objectif est de minimiser le nombre de contraintes liantes tout en imposant une limite sur la taille des cliques. Notre heuristique se présente sous forme d'un Programme en Nombres Entiers (PNE) dans lequel il y a une variable binaire par arête de l'arbre de cliques. Cette variable vaut 1 si l'on fusionne les cliques qui sont reliées par cette arête. On autorise une seule fusion par clique et on autorise seulement les fusions telles que la clique fusionnée ne dépasse pas une certaine taille. L'objectif est de maximiser le nombre de contraintes liantes qui vont disparaître grâce à ces fusions. Plus précisément, le modèle

est le suivant :

$$\begin{aligned} max \quad & 0.5 \sum_{e \in E} d_e(d_e+1)x_e \\ s.c. \quad & |C_e|x_e \leq S^{max} \ \forall e \in E \\ & \sum_{e \in E(i)} x_e \leq 1 \ \forall i \in L \\ & x_e \in \{0,1\} \ \forall e \in E \end{aligned} \tag{4.3}$$

avec

- $U = (L, E)$ l'arbre de cliques ;
- $d_e = |C_i \bigcap C_j|$ le nombre de nœuds en commun pour $e = (C_i, C_j) \in E$;
- $|C_e| = |C_i| + |C_j| - |C_i \bigcap C_j|$ la taille de la clique fusionnée pour $e = (C_i, C_j) \in E$;
- S^{max} un paramètre qui détermine la taille maximale des cliques ;
- $E(i)$ l'ensemble des arêtes qui ont le sommet i comme extrémité.

Cette heuristique, testée pour $S^{max} = 50$ avec MOSEK 8.1, permet de réduire significativement le temps de calcul. On peut la répéter plusieurs fois pour améliorer les performances. Le graphe 4.7 illustre ces améliorations. L'axe des abscisses représente le nombre de fois où l'heuristique a été appliquée. Ainsi, l'abscisse 0 représente la méthode de référence. L'axe des ordonnées représente la somme des temps de résolution MOSEK pour toutes les instances MATPOWER 6.0 (en secondes). On observe, sur ce graphe, une diminution significative du temps de résolution total quand on applique au moins une fois l'heuristique de fusion : on gagne au moins 1000 secondes. D'après nos tests, les meilleures performances sont obtenues après 4 répétitions de l'heuristique (environ 1500 secondes au total). Les tendances observées sur ce graphe se retrouvent instance par instance avec des écarts plus significatifs sur les grosses instances.

Intérêt de la fusion de cliques Les différentes heuristiques de fusion de cliques que nous avons testées ont un impact significatif sur le temps de résolution. De plus, elles permettent de générer des décompositions en cliques assez variées. Par exemple, les décompositions issues de notre heuristique de fusion ont plus de cliques que celles issues de l'heuristique de Molzahn (de l'ordre de 2 fois plus sur les grosses instances). Elles ont aussi beaucoup plus de variables scalaires et de contraintes liantes même si les ordres de grandeur restent similaires. De plus, les décompositions n'ont pas du tout la même répartition en ce qui concerne la taille des cliques comme le montre l'histogramme de la figure 4.8 pour l'instance case9241pegase. Les tailles sont données en variables complexes donc il faut les doubler pour obtenir la taille réelle. Pour l'heuristique de Molzahn, la moitié des cliques est de taille moyenne, entre 10 et

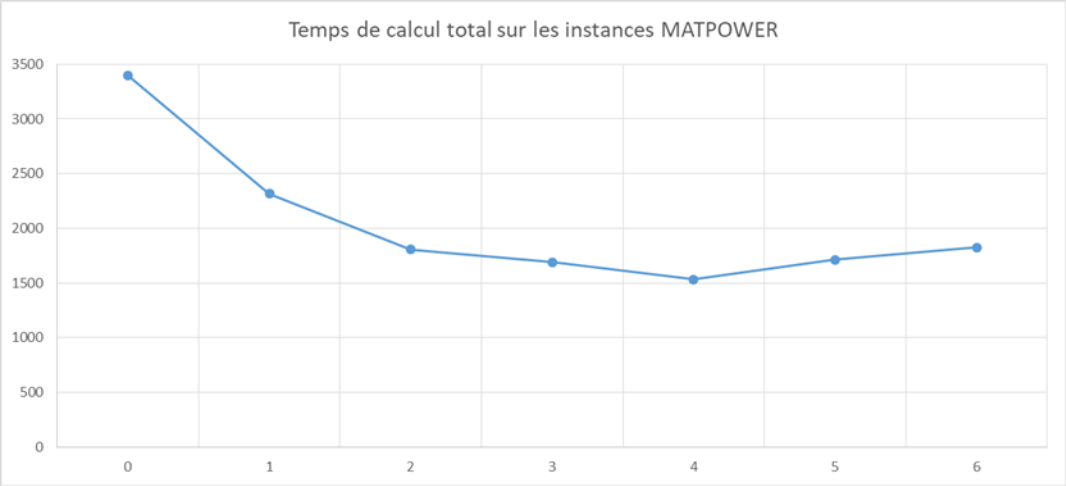

Figure 4.7 Evolution du temps de résolution total sur les instances MATPOWER 6.0 en fonction du nombre de fois où l'heuristique de fusion est appliquée

29 alors que pour la nôtre, 70% des cliques sont de petite taille (taille entre 2 et 9).

Les heuristiques de fusion de cliques semblent donc être une piste intéressante pour accélérer significativement le temps de résolution de la relaxation du rang de l'OPF.

4.4 Proposition d'une nouvelle stratégie de résolution de la relaxation du rang pour l'OPF

Cette section propose une piste de réponse à la question générale soulevée par la section précédente : puisqu'il existe plusieurs décompositions en cliques possibles pour un même problème non équivalentes en terme de résolution par un algorithme de points intérieurs, peut-on trouver celle qui est la plus performante ? Comme nous avons vu que les procédures de fusion de cliques étaient intéressantes pour améliorer les performances, nous proposons une nouvelle heuristique de fusion de cliques.

4.4.1 Estimation du temps d'une itération d'une méthode de points intérieurs

Pour une décomposition donnée, soit $L = \{C_1, ..., C_r\}$ l'ensemble des cliques maximales, m le nombre de contraintes dans le problème SDP original et ℓ le nombre de contraintes liantes déterminé à partir de l'arbre de cliques T. Le cardinal de la clique C_i est noté $|C_i|$. Le temps t d'une itération d'un solveur utilisant une méthode de points intérieurs de pointe est estimé par la formule suivante : $t = \alpha \sum_{i=1}^{r} |C_i|^3 + \beta(m+\ell)^3 + c$ avec α, β et c trois paramètres

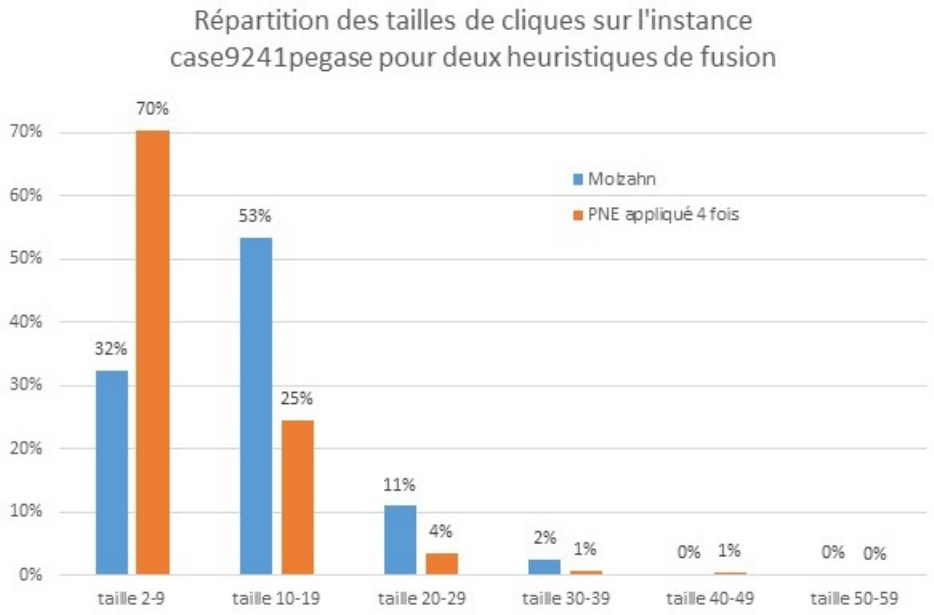

Figure 4.8 Répartition des tailles des cliques pour l'heuristique de Molzahn et pour notre heuristique appliquée 4 fois sur l'instance case9241pegase

inconnus. Cette formule fait intervenir les deux étapes algorithmiques les plus coûteuses d'une méthode de points intérieurs. Le premier terme représente la complexité pour le calcul des valeurs propres pour chaque sous-matrice associée à une clique. Le deuxième terme représente la complexité de la factorisation de Cholesky pour les équations normales.

4.4.2 Détermination des paramètres avec une régression linéaire

Nous déterminons α, β et c pour un problème SDP donné en collectant le temps moyen d'une itération pour différentes décompositions en cliques et en appliquant une régression multilinéaire. Nous utilisons 10 décompositions avec d'importantes différences dans le nombre de cliques et de contraintes liantes pour garantir une certaine diversité. Neuf décompositions ont été obtenues en variables complexes, ce qui produit des cliques en variables complexes. Nous doublons la taille de chaque clique pour obtenir des cliques en variables réelles afin de pouvoir résoudre le problème SDP en variables réelles. La dixième décomposition est calculée directement en variables réelles, en utilisant une factorisation de Cholesky précédée d'un ordre AMD [66]. Parmi les décompositions calculées à partir du problème SDP complexe, les deux premières diffèrent dans le choix de l'algorithme d'extension cordale : une factorisation de Cholesky précédée d'un ordre AMD [66] pour la première et l'heuristique Minimum

Degree [103] pour la deuxième. Les autres décompositions sont obtenues en appliquant des heuristiques de fusion de cliques sur la première décomposition : soit celle décrite dans [1] soit celle dans [106] (présentée dans la section précédente) appliquée une fois ou plus (jusqu'à 6 fois). Tous les problèmes SDP ont été résolus en utilisant MOSEK [100] 9.1. Par ailleurs, nous avons résolu tous les problèmes SDP en multipliant la fonction objectif par 10^{-4} pour améliorer la résolution (*scaling*).

4.4.3 Algorithme proposé

L'heuristique proposée dans [1] cherche à minimiser de manière gloutonne une estimation du coût de résolution d'un problème SDP $f_M(L, \ell)$:

$$f_M(L, \ell) = m + \ell + \sum_{C_i \in L} |C_i|(|C_i| + 1)/2 \tag{4.4}$$

où ℓ dépend de l'arbre de cliques T :

$$\ell = \sum_{(C_i, C_j) \in E} d_{ij}(2d_{ij} + 1) \tag{4.5}$$

avec d_{ij} le nombre de variables complexes partagées par C_i et C_j. Minimiser la fonction $f_M(L, \ell)$, i.e., la taille du problème SDP, entraîne la fusion de petites cliques. Une autre option consiste à minimiser le coût d'une itération de points intérieurs :

$$f(L, \ell) = \alpha \sum_{C_i \in L} |C_i|^3 + \beta(m + \ell)^3 + c \tag{4.6}$$

Cette fonction est une meilleure estimation du coût de résolution d'un problème SDP. Cependant, elle conduit à des fusions de grosses cliques, ce qui est intéressant seulement si l'on fusionne peu de cliques.

Pour tirer parti des deux critères, nous proposons d'appliquer la procédure gloutonne décrite dans l'algorithme 2. L'idée est de fusionner itérativement les deux cliques qui minimisent soit notre critère f soit le critère f_M tant que le nombre de cliques est supérieur à 10% fois le nombre de nœuds. Cette valeur de 10% a été recommandée dans [1]. Nous introduisons un paramètre de changement k_{max} qui indique le nombre d'itérations à partir duquel le critère à minimiser change : au début, le critère à minimiser est f et après k_{max} itérations, le critère à minimiser est f_M. En ajustant k_{max}, nous devrions être capables de calculer des décompositions avec moins de contraintes liantes et des cliques de taille raisonnable. En pratique, nous observons des améliorations du temps de calcul pour des petites valeurs de

k_{max}. Il faut noter que nous avons calculé les deux décompositions (avant et après fusion) en variables complexes. Avant la fusion, nous avons utilisé une factorisation de Cholesky précédée d'un ordre AMD pour calculer l'extension cordale et l'algorithme de Prim pour l'arbre de cliques.

Entrées : Un ensemble de cliques maximales $L = \{C_1, ..., C_r\}$, un arbre de cliques $T = (L, E)$, un paramètre k_{max}

Sorties : Un nouvel ensemble de cliques maximales $L = \{C_1, ..., C_r\}$, un nouvel arbre de cliques $T = (L, E)$

$k = 0$
Calculer ℓ en fonction de T
tant que $|L| \geq 10\%n$ **faire**
    $k \leftarrow k + 1$
    si $k \leq k_{max}$ **alors**
        Trouver l'arête $(C_i, C_j) \in E$ qui minimise $f(L, \ell)$
    sinon
        Trouver l'arête $(C_i, C_j) \in E$ qui minimise $f_M(L, \ell)$
    fin
    $L \leftarrow (L - \{C_i, C_j\}) \cup \{C_i \cup C_j\}$
    Fusionner les nœuds C_i et C_j dans T
    Mettre à jour le nombre de contraintes liantes ℓ
fin

Algorithme 2 : Heuristique de fusion de cliques pour une instance à n nœuds

4.4.4 Résultats numériques

Nous avons testé notre stratégie avec différentes valeurs de k_{max} sur les instances MATPOWER 7.0 [21] de plus de 1000 nœuds (excepté l'instance ACTIVSg70k). Les tests ont été réalisés avec un processeur Intel® Core™ i7-6820HQ CPU @2.70GHz. Nous avons utilisé Julia 1.0.3. [83], JuMP.jl [84], LightGraphs.jl [98], et Mosek.jl avec MOSEK 9.1 [100].

Nous avons comparé nos stratégies à celle présentée dans [1] et à une stratégie basique sans fusion de cliques. Nous avons choisi [1] comme comparateur car c'est l'heuristique la plus rapide qui a été proposée pour les OPF.

Toutes les extensions cordales ont été calculées avec une factorisation de Cholesky précédée d'un ordre AMD. Nous avons divisé la fonction objectif par 10^4 pour un meilleur conditionnement du problème SDP. Nous avons implémenté la stratégie dans [1] dans notre environnement Julia pour une comparaison pertinente. Nous avons testé différentes valeurs de

k_{max}. Nous avons observé une amélioration du temps de calcul pour des petites valeurs de k_{max} : entre 1 et 4. A partir de 5, nous ne constatons plus d'amélioration. L'amélioration la plus significative est obtenue pour $k_{max} = 1$. Les résultats sont présentés dans le graphe de performance dans la Figure 4.9. Ce graphe montre le pourcentage d'instances résolues à chaque seconde pour 4 stratégies : la nôtre avec $k_{max} = 1$ et $k_{max} = 3$, celle dans [1] et pas de fusion de cliques. La plupart des instances sont résolues en moins de 100 secondes pour les trois stratégies de fusion alors que seulement la moitié l'est pour la stratégie basique. Notre stratégie avec $k_{max} = 1$ résout plus d'instances que celle dans [1] à chaque pas de temps. Notre stratégie avec $k_{max} = 3$ semble efficace pour les plus grosses instances qui sont les plus lentes à résoudre. Pour plus de détails, le tableau 4.7 compare les temps de résolution de chaque instance pour les stratégies de fusion avec k_{max} entre 1 et 4. Pour 22 instances sur 23, l'une de nos stratégies est meilleure que celle dans [1]. Toutes les valeurs de k_{max} permettent d'améliorer le temps total de résolution (entre 5% et 12%) mais pour k_{max} entre 2 et 4, l'amélioration est principalement due à l'instance ACTIVSg25k. En effet, les améliorations sont moins significatives sur le temps total de résolution des instances de moins de 20000 nœuds (4% et moins). En revanche, l'amélioration reste importante pour $k_{max} = 1$ (9%). Par ailleurs, l'amélioration moyenne (avec ou sans ACTIVSg25k) est d'au moins 12% pour toutes les valeurs de k_{max}, avec plus de 18% pour $k_{max} = 1$. La stratégie avec $k_{max} = 1$ semble donc être la plus intéressante, d'autant plus qu'elle est la plus stable : elle permet d'améliorer 19 instances sur 23 et ne détériore pas beaucoup le temps de calcul sur les 4 autres instances. En revanche, pour les autres valeurs de k_{max}, nous pouvons constater des améliorations aussi bien que des détériorations importantes. Par exemple, pour $k_{max} = 3$, le temps de calcul est diminué de 21% pour l'instance 6515rte alors qu'il est augmenté de 36% pour les instances 6468rte et 6495rte. Nous recommandons donc d'utiliser notre stratégie avec $k_{max} = 1$. Toutefois, pour les instances de plus de 10000 nœuds, nous recommanderions de tester $k_{max} = 3$ car les améliorations sont plus intéressantes dans ce cas.

Ce travail a été publié dans IEEE Transactions on Power Systems [107].

4.4.5 Stratégie alternative

Nous avons remarqué qu'en utilisant le critère f tout au long de l'algorithme, nous obtenions des décompositions avec des cliques de taille conséquente, ce qui ralentit fortement la résolution (beaucoup de mémoire requise). Pour pallier ce problème, nous avons proposé l'algorithme 2 mais nous avons aussi testé une autre stratégie décrite dans l'algorithme 3. Celle-ci consiste à utiliser le critère f tant que la plus grosse clique ne dépasse pas une certaine taille S_{max} puis à utiliser le critère f_M. Les résultats pour $S_{max} = 40$ et $S_{max} = 50$ sont

Tableau 4.7 Comparaison du temps de résolution MOSEK

Instance	Temps (s)				
	$k_{max} = 1$	$k_{max} = 2$	$k_{max} = 3$	$k_{max} = 4$	[1]
1354pegase	**6.88**	8.22	7.28	7.05	7.19
1888rte	6.8	6.73	6.76	**6.51**	6.59
1951rte	7.06	6.91	6.88	**6.63**	7.55
ACTIVSg2000	**76.59**	88.67	80.56	80.88	80.61
2383wp	**29.86**	35.41	33.95	34.31	37.5
2736sp	49.7	48.48	48.47	**47.45**	64.61
2737sop	47.83	46.08	41.39	**40.63**	56.39
2746wop	72.23	62.83	60.64	**59.27**	68.5
2746wp	49.2	**47.78**	47.89	48.56	56.73
2848rte	**11.22**	12.38	12.28	11.52	14.09
2868rte	**14.3**	15.91	16.02	15.64	18.11
2869pegase	**11.59**	12.63	12.63	12.48	31.17
3012wp	59.88	67.58	65.67	**57.53**	72.5
3120sp	84.06	76	79.23	**71.55**	110.98
3375wp	**57.28**	72.34	69.05	61.19	59.55
6468rte	**93.13**	109.63	164.5	108.22	105.84
6470rte	112.48	119.97	123.2	119.5	**104.55**
6495rte	**99.92**	119.38	183.06	173.06	117.03
6515rte	113	103.7	**100.44**	117	121.91
9241pegase	128.59	131.22	130.58	**122.69**	125.89
ACTIVSg10k	537.63	596.31	**524.05**	548.81	540.3
13659pegase	85.31	82.88	80.7	**79.75**	98.67
ACTIVSg25k	10153.39	10724.7	**10029.44**	10441.2	11377.3
Total	**11907.93**	12595.74	11924.67	12271.43	13283.56
Amélioration par rapport à [1]	12%	5%	11%	8%	
Nb instances améliorées	19/23	14/23	16/23	17/23	
Amélioration moyenne	18.8%	12.9%	13%	17%	
Total sans ACTIVSg25k	1754.54	1871.04	1895.23	1830.23	1906.26
Amélioration par rapport à [1]	9%	2%	1%	4%	
Amélioration moyenne	19.1%	13.2%	13%	17.3%	

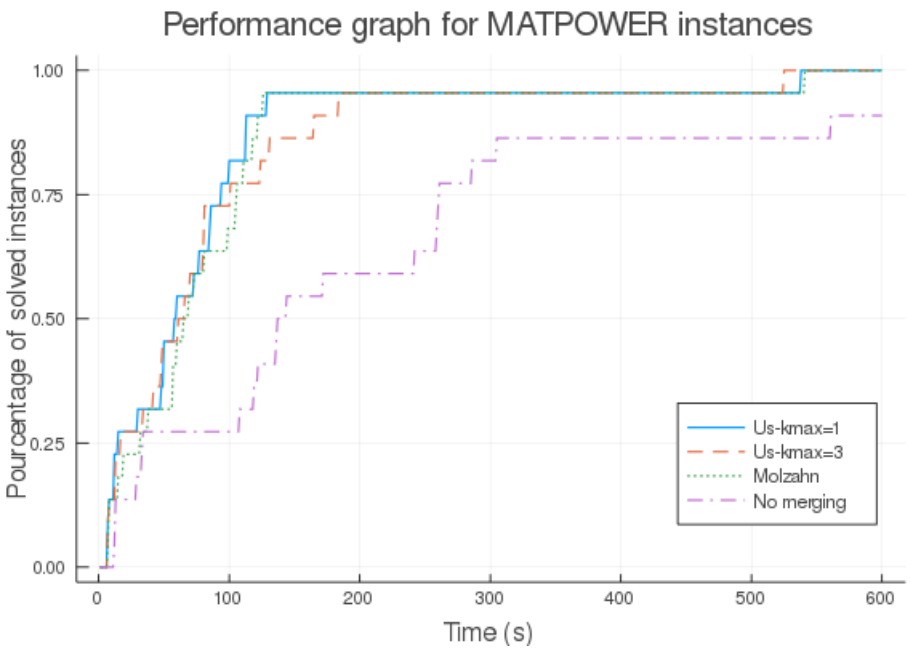

Figure 4.9 Graphe de performance pour les instances MATPOWER de plus de 1000 nœuds (excepté ACTIVSG25k) pour quatre stratégies : la nôtre avec $k_{max} = 1$ et $k_{max} = 3$, celle dans [1] et pas de fusion de cliques.

présentés dans le tableau 4.8. On observe une amélioration de 5% sur le temps de résolution total et de 15% en moyenne pour $S_{max} = 40$. Cette amélioration est cependant moins significative que celle observée avec l'algorithme 2 pour $k_{max} = 1$. Pour $S_{max} = 50$, on observe une détérioration globale assez nette malgré quelques instances améliorées. Il est possible que ces mauvais résultats soient dus à des cliques trop grosses.

tant que $|L| \geq 10\%n$ **faire**
  si $\max_i |C_i| \leq S_{max}$ **alors**
    Trouver l'arête $(C_i, C_j) \in E$ qui minimise $f(L, \ell)$
  sinon
    Trouver l'arête $(C_i, C_j) \in E$ qui minimise $f_M(L, \ell)$
  fin
  $L \leftarrow (L - \{C_i, C_j\}) \cup \{C_i \cup C_j\}$
  Fusionner les nœuds C_i et C_j dans T
  Mettre à jour le nombre de contraintes liantes ℓ
fin

Algorithme 3 : Heuristique alternative de fusion de cliques pour une instance à n nœuds

Tableau 4.8 Comparaison du temps de résolution MOSEK pour une stratégie de fusion de cliques alternative

Instance	$S_{max} = 40$	$S_{max} = 50$	[1]
1354pegase	6.92	**5.81**	7.19
1888rte	7.11	7.44	**6.59**
1951rte	7.74	8.36	**7.55**
ACTIVSg2000	99.91	85.75	**80.61**
2383wp	33.61	**32.11**	37.5
2736sp	**47.3**	48.88	64.61
2737sop	**47.05**	50.66	56.39
2746wop	60.2	**53**	68.5
2746wp	**47.92**	48.81	56.73
2848rte	11.97	**11.75**	14.09
2868rte	16.17	**11.78**	18.11
2869pegase	**12.58**	17.05	31.17
3012wp	**57.95**	65.66	72.5
3120sp	**72.28**	90.31	110.98
3375wp	69	**52.08**	59.55
6468rte	**103.42**	148.92	105.84
6470rte	115.73	145.17	**104.55**
6495rte	**112.83**	153.08	117.03
6515rte	**102.23**	116.03	121.91
9241pegase	121.97	**116.98**	125.89
ACTIVSg10k	662.77	677.22	**540.3**
13659pegase	**79.5**	80.58	98.67
ACTIVSg25k	**10810.83**	67036.55	11377.3
Total	**12706.99**	69063.98	13283.56
Total sans ACTIVSg25k	**1896.16**	2027.43	1906.26
Nombre d'instances améliorées	17/23	15/23	
Amélioration moyenne	15%	7%	

4.5 Extension aux problèmes SDP issus de la hiérarchie de Lasserre d'ordre 2

4.5.1 Décomposition en cliques pour l'ordre 2 de la hiérarchie de Lasserre

La décomposition en cliques maximales est appliquée sur un graphe différent pour les ordres plus grands que 1 de la hiérarchie de Lasserre : le Correlative Sparsity Pattern (CSP) [108] ou graphe de parcimonie corrélatif. Le CSP est défini de la façon suivante : il a pour sommets les variables du problème et il y a une arête entre deux sommets si et seulement si les variables représentées par ces sommets apparaissent simultanément dans une contrainte ou dans un monôme de l'objectif. Pour le problème de l'OPF, le CSP est obtenu à partir du graphe du réseau en reliant par une arête toute paire de sommets à distance au plus 2. Par exemple, la figure 4.10 montre le petit réseau MATPOWER case9 et la figure 4.11 montre le CSP pour ce réseau.

Travailler sur le CSP plutôt que sur le graphe de parcimonie classique (le Sparsity Pattern (SP)) permet d'exprimer chaque contrainte à l'aide d'une seule clique. Toutefois, le CSP est plus dense que le SP pour la relaxation du rang ce qui produit des décompositions en cliques très différentes. Le tableau 4.9 donne une idée plus précise des différences entre les décompositions obtenues sur les deux types de graphes. L'extension cordale est calculée avec une factorisation de Cholesky précédée d'un ordre AMD. Les décompositions contiennent beaucoup de petites cliques avec le SP tandis qu'avec le CSP, les décompositions contiennent moins de cliques mais ces cliques sont plus grosses. Il est aussi intéressant de noter qu'il n'y a qu'une seule clique pour les instances de moins de 9 nœuds avec le CSP, ce qui revient à ne pas décomposer le problème SDP.

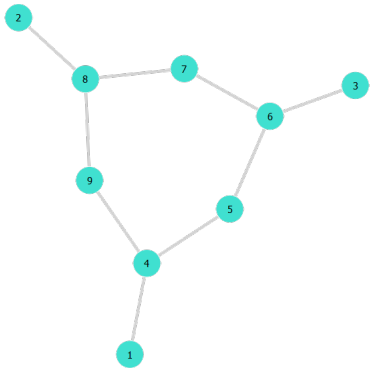

Figure 4.10 Réseau case9

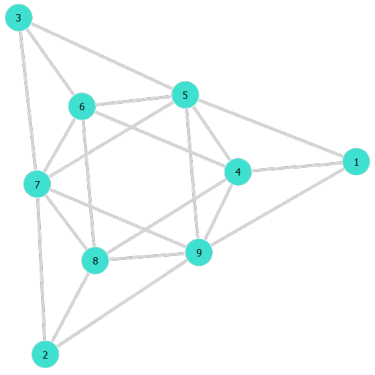

Figure 4.11 CSP case9

Tableau 4.9 Caractéristiques des décompositions en cliques pour le Correlative Sparsity Pattern (CSP) et pour le Sparsity Pattern (SP) - Taille maximale et minimale des cliques données en variables réelles

Instance	CSP		SP	
	#cliques	max/min	#cliques	max/min
WB2	1	4-4	1	4-4
WB5	1	10-10	3	6-6
case6ww	1	12-12	2	8-8
case9	5	10-8	7	6-4
case14	7	14-8	12	6-4
case24_ieee_rts	13	18-8	20	10-4
case30	18	20-8	26	8-4
case39	25	18-6	34	8-4

4.5.2 Impact de la décomposition en cliques sur la résolution

Bien que le graphe de travail soit différent pour les ordres plus élevés de la hiérarchie de Lasserre, nous avons testé l'impact du choix de la décomposition en cliques sur la résolution de l'ordre 2. Nous avons choisi la factorisation de Cholesky précédée d'une permutation AMD comme heuristique d'extension cordale de référence. Nous utilisons le solveur MOSEK version 9.1. pour tous les tests qui suivent. Nous testons d'abord une autre heuristique d'extension cordale, l'heuristique Degré Minimal (MD) décrite dans l'algorithme 1, sur toutes les instances MATPOWER que nous pouvons résoudre à l'ordre 2 (soit jusqu'à 39 nœuds car des problèmes de mémoire apparaissent pour l'instance suivante case57). Pour les plus petites instances, peu de différences sont observées en terme de temps de résolution car les décompositions ont les mêmes caractéristiques. En revanche, des différences peuvent apparaître sur les instances de d'au moins 30 nœuds. L'heuristique MD est notamment intéressante pour l'instance case39 car elle produit une décomposition avec 25 cliques de 6 à 16 nœuds alors que la plus grosse clique avec l'heuristique de référence est de taille 18. Le problème SDP est donc moins gros (30753 contraintes et 95569 variables scalaires au lieu de 32781 contraintes et 101901 variables scalaires) ce qui semble accélérer la résolution (202 secondes au lieu de 262 secondes). D'ailleurs, dans ce cas, l'heuristique MD ajoute moins d'arêtes que l'heuristique de référence AMD+Cholesky : 37 au lieu de 40. C'est le contraire pour l'instance case30, l'heuristique MD ajoute une arête de plus que l'heuristique AMD+Cholesky ce qui augmente la taille du problème SDP. Pourtant la résolution est plus rapide avec l'heuristique MD. Le choix de l'algorithme d'extension cordale semble donc avoir un impact pour les instances de plus de 30 nœuds mais nous avons trop peu de cas tests pour tirer des conclusions.

Nous avons aussi étudié l'impact de la fusion de cliques sur la résolution de l'ordre 2 de

la hiérarchie de Lasserre. En effet, les décompositions en cliques introduisent beaucoup de variables liantes dans le dual du problème des moments (le problème que nous envoyons au solveur). Une façon de diminuer ce nombre de variables est de faire de la fusion de cliques. Or nous ne voulons pas augmenter la taille de la plus grosse clique contrairement à l'ordre 1 car les cliques initiales sont déjà de taille conséquente. Notre algorithme de fusion de cliques PNE, présenté dans la section 4.3.4, est donc pertinent puisqu'une taille maximale est imposée pour les cliques. Nous choisissons de fixer la taille maximale à la taille de la plus grande clique de la décomposition initiale. Nous appliquons cette heuristique de fusion une fois ou deux. Les résultats sont présentés dans le tableau 4.10. Seulement 4 instances sont étudiées car l'algorithme de fusion ne modifie pas la décomposition en-dessous de 14 nœuds et des problèmes de mémoire apparaissent au-dessus de 39 nœuds.

Dans ce tableau de résultats, nous voyons que la fusion de cliques permet effectivement de diminuer le nombre de variables liantes. Cependant, le nombre de variables scalaires total augmente dans la plupart des cas, ce qui conduit à une résolution généralement plus lente. Toutefois, nous n'avons pas assez d'instances pour tirer des conclusions claires et il est possible qu'une fusion de cliques plus progressive soit plus efficace (fusionner deux cliques seulement plutôt que d'autoriser plusieurs fusions de deux cliques).

Pour finir, nous avons aussi testé les différences de résolution entre décomposition en cliques sur le problème réel ou sur le problème complexe. Il semble que ces choix ne soient pas équivalents en terme de résolution mais nous avons trop peu d'instances pour tirer des conclusions claires.

Pour résumer, nous avons observé que le choix de la décomposition en cliques a un impact sur la résolution mais nous pouvons seulement conclure que les observations faites sur l'ordre 1 (la relaxation du rang) ne sont plus valables pour l'ordre 2. Ceci s'explique par le fait que le graphe de départ est différent.

4.5.3 Choix du nœud de référence

Pour l'ordre 1, choisir un nœud de référence n'a pas d'impact sur la résolution (ou alors un léger impact négatif). C'est différent pour l'ordre 2 puisque choisir un bon nœud de référence permet de supprimer beaucoup de termes dans le problème et ainsi de réduire la taille du problème (et potentiellement limiter les problèmes numériques). Nous avons testé plusieurs choix raisonnables pour le nœud de référence : choisir le nœud le plus maillé dans le réseau de transport, le plus maillé dans le CSP ou encore le plus maillé dans l'extension cordale du CSP. Les résultats sont présentés dans le tableau 4.11. Quelques variations sont observées dans les temps de résolution lorsque les nœuds choisis sont différents. Malgré le peu de cas

Tableau 4.10 Test fusion de cliques PNE avec S^{max} = taille max de la décomposition initiale (les tailles max/min dans le tableau sont données en terme de variables réelles)

Instance/Algo	#cliques	max/min	#var liantes	#var scalaires	Temps(s)	Statut
case14						
Pas de fusion	7	14-8	3475	29012	16.86	SP
Fusion PNE×1	6	14-8	2705	27920	20.81	OPT
case24_ieee_rts						
Pas de fusion	13	18-8	14174	109037	390.03	OPT
Fusion PNE×1	10	18-8	11127	106150	367.17	OPT
Fusion PNE×2	9	18-8	10761	116169	478.75	OPT
case30						
Pas de fusion	18	20-8	19600	143414	925.47	OPT
Fusion PNE×1	12	20-8	14359	147604	969.17	OPT
Fusion PNE×2	11	20-8	14212	153317	1099.02	SP
case39						
Pas de fusion	25	18-6	13600	101901	262.41	SP
Fusion PNE×1	14	18-6	8843	120013	306.94	SP
Fusion PNE×2	11	18-12	8285	137697	507.80	SP

tests, il semble quand même intéressant de choisir le nœud le plus maillé dans l'extension cordale du CSP pour que le problème soit le plus petit possible.

4.5.4 Génération de contraintes et colonnes pour la hiérarchie de Lasserre

La hiérarchie de Lasserre est une approche très limitée en pratique car les problèmes SDP issus de cette hiérarchie grossissent très vite lorsque l'on augmente l'ordre. L'ordre 2 peut déjà être problématique comme mentionné précédemment : des problèmes numériques apparaissent sur les petites instances MATPOWER et nous n'arrivons pas à résoudre une instance plus grande

Tableau 4.11 Test de trois nœuds de référence : le plus maillé dans le réseau, le plus maillé dans le CSP, le plus maillé dans l'extension cordale du CSP (H CSP)

Instance	Choix du nœud	Nœud	#ctr	#var	Temps (s)	Statut
case9	réseau	4	2050	6924	1.03	OPT
case9	CSP/H CSP	5	1922	6442	0.98	OPT
case14	réseau/CSP	4	9456	29012	16.86	SP
case14	H CSP	5	8456	25802	14.77	SP
case24_ieee_rts	réseau/CSP/H CSP	9	33225	109037	390.03	OPT
case30	réseau/CSP	6	45948	143414	925.47	OPT
case30	H CSP	4	45244	141298	674.00	SP
case39	réseau/CSP/H CSP	16	32781	101901	262.41	SP

que l'instance case39. Une approche de hiérarchie de Lasserre partielle a été proposée pour éviter que les relaxations ne grossissent trop vite : l'idée est de construire des relaxations SDP entre deux ordres pour tirer parti de la précision d'un ordre plus élevé et du coût faible d'un ordre plus bas [7, 80]. Nous voulions explorer une autre approche : utiliser une méthode de décomposition pour augmenter progressivement la difficulté du problème. Plus précisément, nous voulions nous inspirer d'une méthode de génération de contraintes comme celle proposée dans [19] pour résoudre la relaxation SDP d'ordre 2. Notre idée consistait à ajouter progressivement les contraintes liées aux moments d'ordre 2 dans le problème dual. Cependant, si l'on raisonne uniquement sur un certain type de variables ou de contraintes, le problème reste de taille conséquente et l'on ne gagne quasiment rien en terme de résolution. Il faut donc raisonner de manière plus globale : soit toutes les variables et toutes les contraintes associées à une clique sont à l'ordre 1 soit elles sont à l'ordre 2. Raisonner comme cela permet de diminuer la taille des problèmes lorsque peu de cliques sont à l'ordre 2. Si toutes les cliques sont à l'ordre 1, le problème obtenu est la relaxation du rang (l'ordre 1). Mais en pratique cet ordre 1 se résout très mal : la résolution est lente et des problèmes numériques sont observés même pour des petites instances. Ceci s'explique par le fait que les cliques proviennent d'une décomposition sur le CSP et que la décomposition obtenue n'est pas très pertinente pour l'ordre 1. De plus, résoudre le problème dual empire les choses dans ce cas : pour la relaxation du rang, il vaut mieux résoudre le problème primal, pour les ordres plus élevés, le problème dual. L'initialisation étant déjà problématique, une approche génération de colonnes et/ou contraintes n'est pas viable. Une alternative consiste à redéfinir les cliques à chaque itération pour pouvoir traiter des problèmes un peu plus grands. Toutefois, ce type d'approche nous fait sortir du cadre de la génération de colonnes et/ou contraintes car le problème à résoudre évoluerait au cours du temps et il faudrait certainement choisir de manière heuristique les cliques qui passent à l'ordre 2. Ce type d'approche serait en réalité très similaire à celle proposée par Josz et Molzahn dans [7].

4.6 Conclusion et perspectives

Dans ce chapitre, nous nous sommes concentrés sur la résolution de la relaxation du rang de l'OPF. Nous avons d'abord montré qu'il est pertinent de travailler sur le choix de la décomposition en cliques maximales pour accélérer la résolution. Nous avons ensuite proposé une nouvelle stratégie de fusion de cliques qui améliore significativement les performances obtenues avec la meilleure heuristique de fusion de cliques connue pour les problèmes d'OPF. Ces travaux ont été publiés dans le journal IEEE Transactions on Power Systems [107]. Pour finir, nous avons cherché à accélérer la résolution de la relaxation d'ordre 2 de la

hiérarchie de Lasserre. Nous avons d'abord exploré plusieurs choix de décompositions en cliques maximales mais aucune stratégie n'a été mise en évidence. Nous avons aussi essayé sans succès de concevoir une méthode de génération de colonnes et contraintes.

Nos travaux sur la résolution de la relaxation du rang de l'OPF pourraient être continués en s'orientant vers des méthodes d'intelligence artificielle pour apprendre les caractéristiques d'une bonne décomposition en cliques maximales. Utiliser des techniques de réduction faciale après notre décomposition en cliques maximales pourrait être une autre piste d'amélioration [17]. En ce qui concerne l'ordre 2 de la hiérarchie de Lasserre, d'autres méthodes de décomposition comme la méthode ADMM [72] pourraient être plus pertinentes.

CHAPITRE 5 RÉSOLUTION DE PROBLÈMES D'OPF RÉACTIF VIA LA PROGRAMMATION SEMI-DÉFINIE POSITIVE

Dans ce chapitre, nous nous intéressons au problème de l'OPF Réactif (ROPF) qui correspond à un problème d'OPF avec des moyens d'action supplémentaires. Nous proposons des nouvelles versions de ce problème, plus proches des problématiques de RTE. Nous les résolvons à l'aide d'une méthode d'énumération implicite qui s'appuie sur une relaxation SDP et nous montrons qu'une telle méthode est plus précise et plus robuste qu'une méthode d'arrondi.

5.1 Motivations

Le problème ROPF est un problème qui modélise un peu plus finement le fonctionnement réel du réseau puisqu'il prend en compte des moyens d'action qui ne sont pas présents dans la formulation classique du problème de l'OPF. Le problème de l'OPF prend en compte le premier moyen de réglage de la tension sur le réseau : le réglage de tension des groupes. Deux moyens d'action supplémentaires pour le réglage de la tension sont modélisés dans le problème ROPF classique : la mise en service ou hors service des MCS et le réglage des ratios de transformation des transformateurs. La prise en compte de ces moyens d'action implique l'utilisation de variables discrètes. Le problème ROPF est ainsi un problème non convexe, non linéaire et en variables mixtes (MINLP non convexe). Les problèmes de ce type sont NP-difficiles et il est difficile de calculer un optimum global en pratique. Dans certains cas, il peut même s'avérer difficile de trouver une solution réalisable. C'est pourquoi beaucoup de travaux de recherche sur le problème ROPF se concentrent sur le calcul d'une solution réalisable, e.g. avec des heuristiques stochastiques [109–116] ou avec des méthodes d'intelligence artificielle [117]. D'autres travaux utilisent une relaxation convexe pour obtenir une borne inférieure mais aussi pour calculer une solution réalisable avec une heuristique d'arrondi : [118] propose une relaxation SDP, [119] une relaxation SOCP et [120] une relaxation quadratique qui n'est pas appliquée au problème ROPF mais qui pourrait l'être. Des techniques d'arrondi sont aussi proposées dans [121] et [122]. Récemment, [123] utilise un modèle qui ne contient que des variables continues et propose une nouvelle formulation SDP. Un autre travail récent [124] propose d'utiliser une relaxation SDP avec des variables entières et présente un algorithme d'énumération implicite pour résoudre cette relaxation en variables mixtes. Dans ce travail, une modélisation discrète des MCS et des prises des transformateurs est utilisée. Des solutions réalisables sont calculées grâce à des solveurs MINLP sans utiliser l'information

de la relaxation SDP en variables mixtes. Cette méthode de résolution est testée sur des petits réseaux (jusqu'à 300 nœuds). Finalement, d'autres méthodes comme des méthodes d'énumération implicite spatiale (*spatial Branch-and-Bound*) pourraient être utilisées. Voir [53] pour un résumé des différentes méthodes pour les MINLP non convexes. Cependant, la modélisation classique du problème ROPF n'est pas forcément la plus pertinente du point de vue d'un gestionnaire de réseau comme RTE. D'abord, le ROPF n'associe pas de coût à la mise en service ou hors service des MCS. Or en pratique ce sont des éléments fragiles qui peuvent être rapidement détériorés s'ils sont constamment manipulés. Ensuite, pour des problèmes à court terme, le plan de production est défini à l'avance et il n'est possible de le modifier que sous certaines conditions. Finalement, le problème ROPF modélise la variation du ratio de transformation des transformateurs avec des variables discrètes mais les nouvelles technologies de transformateur permettent une modélisation continue. Contrairement au problème ROPF classique de la littérature, il nous semble plus intéressant de nous concentrer sur les MCS dans un premier temps (en laissant les ratios de transformation des transformateurs fixes). Idéalement, nous devrions prendre en compte tous les moyens d'action qui existent pour le réglage de la tension (Compensateurs Statiques de Puissance Réactive (CSPR), ouverture de lignes, etc) mais nous préférons limiter le nombre de moyens d'action sur la tension et être capable de calculer de bonnes solutions réalisables dans une limite de temps raisonnable plutôt que d'avoir un problème complet pour lequel nous ne pouvons pas calculer de bonnes solutions. Les deux motivations principales de ce chapitre sont les suivantes : 1. définir des variantes du problème ROPF qui sont plus adaptées aux problématiques de RTE ; 2. proposer une méthode d'énumération implicite qui s'appuie sur une relaxation SDP pour résoudre ces problèmes. A notre connaissance, c'est la première fois qu'une telle méthode est utilisée pour le problème ROPF.

5.2 Modélisation

Le problème ROPF est une généralisation du modèle (OPF) présenté dans la section 2.2. La modélisation qui nous intéresse fait intervenir une nouvelle catégorie de variables : les variables binaires u_s qui définissent, pour chaque MCS $s \in S$, s'il est en service ou non. Nous introduisons aussi trois contraintes optionnelles.

La première contrainte optionnelle, notée MAXkshunts, limite le nombre de MCS qui peuvent être mis en service. Cette contrainte est une manière de répondre à la problématique des coûts de mise en service des MCS. En effet, le problème ROPF classique n'attribue aucun coût à la mise en service ou à la mise hors service d'un MCS. En pratique, le coût n'est pas nul car les MCS sont des équipements qu'il faut enclencher manuellement et il n'est pas souhaitable de

les manipuler constamment. Introduire des coûts artificiels serait une option pour répondre à cette problématique mais il est difficile de définir des coûts pertinents. Nous préférons donc utiliser une alternative qui consiste à autoriser $\mathbf{k}$ mises en service de MCS au maximum :

$$\sum_{n \in S} u_n \leq \mathbf{k}. \qquad \text{(MAXkshunts)}$$

Nous raisonnons ici dans l'absolu c'est-à-dire que nous n'envisageons pas de situation initiale particulière (ce qui est aussi le cas dans le problème ROPF classique). Si aucune contrainte n'est imposée sur les MCS et qu'ils ne possèdent pas de coût de mise en service, l'optimiseur a tendance à mettre en service la plupart des MCS. Cependant, il existe souvent des solutions qui contiennent peu de MCS en service pour lesquelles la valeur optimale du problème n'est pas détériorée significativement. L'ajout de la contrainte (MAXkshunts) permet de trouver de telles solutions.

Nous utilisons la valeur $\mathbf{k} = 4$ par défaut car c'est une valeur utilisée chez RTE pour limiter l'utilisation d'une autre manoeuvre préventive : les parades topologiques. En effet, dans les études long terme, il existe un critère qui limite le nombre de changements topologiques à 4 par jour, afin de traduire les contraintes d'exploitabilité liées à la mise en oeuvre de changements de topologie fréquents. Ce critère permet de ne pas occulter de réelles limites du réseau derrière une optimisation irréaliste des capacités d'exploitation. Nous choisissons donc d'utiliser la même valeur pour limiter le nombre de MCS en service mais cette valeur est arbitraire et pourrait être adaptée en fonction des données métier. Si $\mathbf{k} = 4$ implique un problème non réalisable (prouvé ou supposé), ce qui peut arriver pour des réseaux de grande taille, nous définissons $\mathbf{k}$ comme le plus petit entier pour lequel il est possible de calculer une solution réalisable.

La deuxième contrainte optionnelle, notée MAXkmoves, concerne aussi les MCS mais répond à une problématique plus court terme. Il est en effet préférable de limiter les mouvements des MCS par rapport à l'état initial du réseau (qui est connu à court terme) car un dispatcher (personne en charge du pilotage des flux électriques) ne peut effectuer qu'un nombre d'actions limité en un temps limité et il y a toujours une problématique d'usure. A court terme, nous connaissons l'état $\mathbf{u^0}$ des MCS ($\mathbf{u^0_s} = 1$ si le MCS s est en service et inversement). Nous ajoutons donc une contrainte qui tient compte de cet état et qui impose $\mathbf{k}$ changements maximum par rapport au vecteur $\mathbf{u^0} = 1$. Ce type de contrainte est appelé contrainte de branchement dans le domaine de l'optimisation linéaire en variables mixtes [125] et elle s'exprime sous la forme suivante :

$$\sum_{n \in S : \mathbf{u_n^0}=0} u_n + \sum_{n \in S : \mathbf{u_n^0}=1} (1 - u_n) \leq \mathbf{k}. \qquad \text{(MAXkmoves)}$$

Nous utilisons encore la valeur $\mathbf{k} = 4$ et ce, pour les mêmes raisons que précédemment.

La troisième contrainte optionnelle, notée GENmoves, concerne la production active et des problèmes très court terme. Cette contrainte a pour objectif de simuler le réglage primaire de fréquence en autorisant une baisse ou une augmentation coordonnée de production par rapport à un plan de production initial $\mathbf{P^0}$. Plus précisément, deux variables binaires δ^+ et δ^- indiquent si le plan de production est augmenté ou diminué. Ces deux variables sont incompatibles : si $\delta^+ = 1$ alors $\delta^- = 0$ et vice versa (ce qui est imposé par une contrainte sur leur somme). Ceci signifie que tous les générateurs vont dans le même sens pour compenser un manque ou un excès de production. De plus, chaque générateur doit contribuer au pro rata de sa capacité. Pour ce faire, nous utilisons une contrainte affine avec une unique variable λ^+ ou λ^- qui implique que tous les générateurs atteignent leur borne supérieure (ou inférieure) au même moment ($\lambda^+ = 1$ ou $\lambda^- = 0$). Les variables λ^+ et λ^- ont des bornes telles que, pour chaque générateur $n \in G$, la puissance active produite $Re(S_n)$ soit toujours entre ses bornes $\mathbf{P_n^{min}}$ et $\mathbf{P_n^{max}}$. Ainsi, si les contraintes de GENmoves sont utilisées, les contraintes $\mathbf{P_n^{min}} \leq Re(S_n) \leq \mathbf{P_n^{max}}$ sont redondantes.

Pour résumer, l'ajout de GENmoves dans le problème ROPF implique l'utilisation de quatre variables additionnelles dont deux binaires et contraint la génération de puissance active par rapport à un plan de production initial. L'ensemble des contraintes de GENmoves est détaillé ci-dessous :

$$\begin{cases} Re(S_n) = [\mathbf{P_n^{min}} + 2(\mathbf{P_n^0} - \mathbf{P_n^{min}})\lambda^-]\delta^- \\ +[2\mathbf{P_n^0} - \mathbf{P_n^{max}} + 2(\mathbf{P_n^{max}} - \mathbf{P_n^0})\lambda^+]\delta^+ \quad \forall n \in G \\ 0 \leq \lambda^- \leq 0.5 \\ 0.5 \leq \lambda^+ \leq 1 \\ \delta^+ + \delta^- = 1 \\ \lambda^-, \lambda^+ \in \mathbb{R} \\ \delta^+, \delta^- \in \{0, 1\}. \end{cases} \qquad \text{(GENmoves)}$$

Il faut noter que cette modélisation du réglage primaire de fréquence pourrait être raffinée puisqu'en pratique les coefficients de participation des groupes à une baisse ou une montée du plan de production sont différents et qu'il faudrait prendre en compte le fait que certains groupes arrivent en butée avant d'autres. Cependant, une telle modélisation complexifierait encore le problème et nous préférons nous contenter d'une modélisation plus grossière.

Le modèle du problème ROPF qui nous intéresse est donc le suivant :

$$
\begin{array}{lll}
\min\limits_{v,S,u} & \sum\limits_{g\in G} \mathbf{c_g}(Re(S_g)) + \mathbf{k_g} & \\
s.c. & S_n = \mathbf{S_n^l} + (\mathbf{g_n} - \mathbf{jb_n})|v_n|^2 u_n + \sum\limits_{l=(n,d)} (\frac{\overline{\mathbf{y_l}} - \mathbf{jb_l}}{\boldsymbol{\tau}_\mathbf{l}^\mathbf{2}}|v_n|^2 - \overline{\mathbf{y_l}}\frac{\mathbf{e}^{\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} v_n \overline{v_d}) & \\
& - \sum\limits_{l=(o,n)} (-\overline{\mathbf{y_l}}\frac{\mathbf{e}^{-\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}}\overline{v_o} v_n + (\overline{\mathbf{y_l}} - \mathbf{jb_l})|v_n|^2) & \forall n \in N \\
& \mathbf{P_n^{min}} \leq Re(S_n) \leq \mathbf{P_n^{max}} & \forall n \in G \\
& \mathbf{Q_n^{min}} \leq Im(S_n) \leq \mathbf{Q_n^{max}} & \forall n \in G \\
& S_n = 0 & \forall n \notin G \\
& (\mathbf{v_n^{min}})^2 \leq |v_n|^2 \leq (\mathbf{v_n^{max}})^2 & \forall n \in N \\
& |\frac{\overline{\mathbf{y_l}} + \mathbf{jb_l}}{\boldsymbol{\tau}_\mathbf{l}^\mathbf{2}} v_{o_l} - \overline{\mathbf{y_l}}\frac{\mathbf{e}^{-\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} v_{d_l}|^2 \leq (\mathbf{i_l^{max}})^2 & \forall l \in B \\
& |-\overline{\mathbf{y_l}}\frac{\mathbf{e}^{\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} v_{o_l} + (\overline{\mathbf{y_l}} + \mathbf{jb_l}) v_{d_l}|^2 \leq (\mathbf{i_l^{max}})^2 & \forall l \in B \\
& \text{(MAXkshunts) OR (MAXkmoves) OR (GENmoves)} & \\
& u_n = 0 & \forall n \notin S \\
& v_n \in \mathbb{C}, S_n \in \mathbb{C} & \forall n \in N \\
& u_n \in \{0,1\} & \forall n \in S.
\end{array}
\tag{ROPF}
$$

5.3 Relaxation SDP

Dans cette section, nous présentons la relaxation SDP utilisée pour calculer des bornes inférieures du problème (ROPF) dans notre procédure d'énumération implicite.

Pour construire une relaxation SDP du problème ROPF, nous introduisons la matrice Hermitienne $V = vv^H$ que nous relâchons en $V \succeq 0$. Pour les variables binaires, nous relâchons les variables binaires u_i et nous introduisons de nouvelles variables ξ_i pour modéliser le produit $u_i V_{ii}$. Les contraintes quadratiques $\xi_i = u_i V_{ii}$ sont alors relâchées en utilisant les enveloppes de McCormick [126]. Pour un produit $w = xy$ avec $x \in [\underline{x}, \overline{x}]$ et $y \in [\underline{y}, \overline{y}]$, les enveloppes de McCormick sont détaillées dans l'équation (McC).

$$
w = xy \Rightarrow \begin{cases} w \leq \overline{x}y + x\underline{y} - \overline{x}\underline{y} \\ w \leq x\overline{y} + \underline{x}y - \underline{x}\overline{y} \\ w \geq \underline{x}y + x\underline{y} - \underline{x}\underline{y} \\ w \geq \overline{x}y + x\overline{y} - \overline{x}\overline{y} \end{cases}
\tag{McC}
$$

Pour chaque MCS $i \in S$, $u_i \in [0,1]$ et $V_{ii} \in [(\mathbf{v_i^{min}})^2, (\mathbf{v_i^{max}})^2]$, donc les enveloppes de McCormick pour le produit $\xi_i = u_i V_{ii}$ sont :

$$\begin{cases} \xi_i \leq V_{ii} + (\mathbf{v_i^{min}})^2(u_i - 1) \\ \xi_i \leq (\mathbf{v_i^{max}})^2 u_i \\ \xi_i \geq (\mathbf{v_i^{max}})^2(u_i - 1) + V_{ii} \\ \xi_i \geq (\mathbf{v_i^{min}})^2 u_i \end{cases} . \qquad (\text{McC}_i)$$

La relaxation SDP obtenue est la suivante :

$$\begin{array}{lll} \min\limits_{V,S,u,\xi} & \sum\limits_{g\in G} \mathbf{c_g}(Re(S_g)) + \mathbf{k_g} & \\ s.c. & S_n = \mathbf{S_n^l} + (\mathbf{g_n} - \mathbf{jb_n})\xi_n + \sum\limits_{l=(n,d)} (\frac{\overline{\mathbf{y_l}} - \mathbf{jb_l}}{\tau_\mathbf{l}^\mathbf{2}} V_{nn} - \overline{\mathbf{y_l}} \frac{\mathbf{e}^{\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} V_{nd}) & \\ & - \sum\limits_{l=(o,n)} (-\overline{\mathbf{y_l}} \frac{\mathbf{e}^{-\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} V_{no} + (\overline{\mathbf{y_l}} - \mathbf{jb_l}) V_{nn}) & \forall n \in N \\ & \mathbf{P_n^{min}} \leq Re(S_n) \leq \mathbf{P_n^{max}} & \forall n \in G \\ & \mathbf{Q_n^{min}} \leq Im(S_n) \leq \mathbf{Q_n^{max}} & \forall n \in G \\ & (\mathbf{v_n^{min}})^2 \leq V_{nn} \leq (\mathbf{v_n^{max}})^2 & \forall n \in N \\ & I_l^{orig} \cdot V \leq (\mathbf{i_l^{max}})^2 & \forall l \in B \\ & I_l^{dest} \cdot V \leq (\mathbf{i_l^{max}})^2 & \forall l \in B \\ & S_n = 0 & \forall n \notin G \\ & (\text{MAXkshunts}) \ \mathsf{OR} \ (\text{MAXkmoves}) \ \mathsf{OR} \ (\text{GENmoves}) & \\ & (\text{McC}_i) & \forall i \in S \\ & u_n = 0 & \forall n \notin S \\ & \xi_n = 0 & \forall n \notin S \\ & V \succeq 0 & \\ & S_n \in \mathbb{C} & \forall n \in N \\ & u_n \in [0, 1] & \forall n \in S \end{array} \qquad (5.1)$$

dans laquelle pour chaque ligne $l \in B$, I_l^{orig} (resp. I_l^{dest}) est la matrice Hermitienne telle que $I_l^{orig} \cdot vv^H = |\frac{\mathbf{y_l} + \mathbf{jb_l}}{\tau_\mathbf{l}^\mathbf{2}} v_{o_l} - \mathbf{y_l} \frac{\mathbf{e}^{-\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} v_{d_l}|^2$ (resp. $I_l^{dest} \cdot vv^H = |-\mathbf{y_l} \frac{\mathbf{e}^{\mathbf{j}\theta_\mathbf{l}}}{\tau_\mathbf{l}} v_{o_l} + (\mathbf{y_l} + \mathbf{jb_l}) v_{d_l}|^2$).

Cette relaxation SDP en variables complexes peut être convertie en variables réelles en utilisant la représentation rectangulaire et une matrice de taille $2n$. De plus, elle peut être reformulée en utilisant une décomposition en cliques pour accélérer la résolution.

Nous avons essayé d'améliorer cette relaxation en testant deux options sur le problème ROPF avec la contrainte MAXkshunts : la quadratisation de la contrainte MAXkshunts et une procédure de contraction de bornes, appelée OBBT (*Optimality-Based Bound Tightening* en anglais).

Quadratisation de la contrainte MAXkshunts Nous introduisons la variable $U = \begin{pmatrix} 1 & u^T \\ u & uu^T \end{pmatrix}$ que nous relâchons en $U \succeq 0$. La relaxation SDP est alors modifiée de la façon suivante :

- Les variables u_n sont remplacées par la matrice semi-définie positive U de taille $(|S|+1)$ pour laquelle $U_{11} = 1$ et $U_{ii} = U_{1i}\ \forall i \geq 2$;
- La contrainte linéaire $\sum_{n \in S} u_n \leq k$ est remplacée par $2|S|$ contraintes :

$$\sum_{i \in S} U_{ij} \leq k U_{1j}\ \forall j \in S$$

et

$$\sum_{i \in S} (U_{1i} - U_{ij}) \leq k(1 - U_{1j})\ \forall j \in S$$

(ce qui revient à multiplier la contrainte linéaire par u_j et $(1 - u_j)\ \forall j \in S$).

Cette quadratisation a un effet négligeable sur la valeur optimale de la relaxation SDP.

Contraction de bornes (OBBT) L'idée d'une procédure OBBT [127] est de trouver les bornes les plus serrées possibles pour certaines variables. Ceci peut être très utile dans le cas d'une procédure d'énumération implicite spatiale. Dans notre cas, nous cherchons à améliorer les enveloppes de McCormick grâce à de meilleures bornes sur le module de la tension. Pour cela, nous résolvons $2|S|$ problèmes SDP (deux pour chaque nœud qui possède un MCS). Ces problèmes ont la forme suivante :

$$\begin{array}{ll} \min\limits_{V,S,u,\xi} / \max\limits_{V,S,u,\xi} & V_{kRe,kRe} + V_{kIm,kIm} \\ s.c. & \mathsf{contraintes\ de\ la\ relaxation\ SDP} \\ & \mathsf{objectif\ de\ la\ relaxation\ SDP} \leq \mathsf{UB}. \end{array} \tag{5.2}$$

L'ajout de la dernière contrainte est conseillé dans la thèse d'Hadrien Godard [8]. Cette contrainte, utilisant une borne supérieure UB de notre MINLP, permet de restreindre le domaine réalisable aux solutions qui fournissent une borne inférieure.

Nous testons deux choix pour la valeur UB : la meilleure borne supérieure connue pour notre MINLP et cette même borne diminuée de 10^{-4} de sa valeur. Ces deux options permettent d'améliorer les bornes sur le module de la tension pour tous les nœuds avec MCS. La deuxième option les améliore légèrement plus. Cependant, dans les deux cas, la valeur optimale de la relaxation SDP avec ces nouvelles bornes est quasiment la même que celle avec les bornes originales.

Nous utilisons donc la relaxation SDP définie dans 5.1 dans toutes nos expérimentations.

5.4 Résolution

Dans cette section, nous présentons notre méthode de résolution pour les trois variantes du problème ROPF. Dans un premier temps, nous calculons un encadrement de la valeur optimale grâce à notre relaxation SDP et à un solveur local non linéaire. Ensuite, si l'encadrement n'est pas satisfaisant, nous appliquons une procédure d'énumération implicite.

5.4.1 Calcul d'un encadrement de la valeur optimale

Nous pouvons déjà calculer une borne inférieure, notée LB, en résolvant la relaxation SDP présentée dans la section 5.3. Nous utilisons une décomposition en cliques maximales pour accélérer la résolution. Pour les instances de moins de 1000 nœuds, nous utilisons une décomposition classique calculée à partir d'une factorisation de Cholesky précédée d'un ordre AMD [66]. Pour les instances de plus de 1000 nœuds, nous utilisons la décomposition obtenue en appliquant l'algorithme de fusion de cliques présenté dans la section 4.4 avec le paramètre $k_{max} = 1$.

Nous calculons une borne supérieure, notée UB, en utilisant un solveur local non linéaire et une procédure en trois étapes. D'abord, nous résolvons la relaxation continue du problème ROPF. Ceci permet de trouver un bon point de départ pour résoudre le problème en variables mixtes. Ensuite, nous résolvons le problème en variables mixtes en utilisant une reformulation MPEC (Programmation Mathématique avec des Contraintes d'Equilibre), c'est-à-dire que les contraintes de binarité ($x \in \{0, 1\}$) sont remplacées par des contraintes de complémentarité ($x(x-1) = 0$) qui sont ensuite ajoutées à la fonction objectif avec un terme de pénalisation. Ce terme est automatiquement mis à jour au cours de la procédure par le solveur Knitro que nous utilisons pour la résolution locale non linéaire. Finalement, nous fixons les variables binaires en arrondissant la solution obtenue et nous résolvons le problème continu qui en résulte. Cette dernière phase permet de s'assurer de la réalisabilité de la solution.

Le saut d'optimalité est alors donné par la formule $\frac{UB-LB}{UB}$. Nous l'exprimons habituellement en pourcentage. Nous considérons que le saut d'optimalité est satisfaisant s'il est strictement inférieur à 10^{-4}, ce qui est une valeur satisfaisante pour RTE. Si le saut d'optimalité obtenu est supérieur ou égal à cette valeur, nous appliquons la procédure d'énumération implicite décrite dans la sous-section suivante.

5.4.2 Énumération implicite

Nous proposons d'utiliser une procédure d'énumération implicite qui utilise la relaxation SDP présentée dans la section 5.3 pour calculer une borne inférieure à chaque nœud. Cette procédure est décrite en détail dans l'algorithme 4. Cet algorithme prend en entrée une instance, la solution de la relaxation SDP (équation 5.1) et deux seuils l et u entre 0 et 1. Il retourne une borne supérieure de la valeur optimale du problème. La première étape de cet algorithme (ligne 1) consiste à fixer les variables binaires pour lesquelles la valeur de la relaxation SDP est supérieure au seuil u et inférieure au seuil l. Ceci définit un premier dictionnaire de variables fixées (ligne 2). Une première borne supérieure est alors calculée à partir de ce dictionnaire de variables fixées (ligne 3). La fonction `solve_MINLP` applique la procédure en trois étapes décrite dans la sous-section précédente pour calculer une solution réalisable. Si toutes les variables binaires sont fixées, seule la troisième étape de la procédure est réalisée. La ligne suivante (ligne 4) initialise la procédure d'énumération implicite. Nous utilisons une structure `node` qui est définie par la borne inférieure du nœud père et le dictionnaire de variables binaires fixées à ce nœud. Nous définissons `nodes_list` comme la liste des nœuds à explorer. Le premier nœud de cette liste est défini par une borne inférieure père qui vaut $-\infty$ et le premier dictionnaire de variables fixées. Tant qu'il y a des nœuds à explorer (tant que `nodes_list` est non vide), la procédure est la suivante. Un nœud est sélectionné selon la stratégie profondeur d'abord en privilégiant le nœud avec le plus de variables fixées à 1 (ligne 6). Le dictionnaire des variables binaires fixées à ce nœud est utilisé pour les calculs suivants (ligne 7) et le nœud est détruit (ligne 8). La fonction `solve_SDP` calcule une borne inférieure de la valeur optimale en utilisant la relaxation SDP (équation 5.1) selon le dictionnaire de variables fixées. Ensuite, il y a trois cas possibles :

- (lignes 10 :11) Si la borne inférieure est strictement supérieure à la borne supérieure courante (cette condition peut être relâchée à la précision demandée) ou si la relaxation SDP est non réalisable, le nœud est coupé, i.e., nous n'explorons pas les nœuds-fils de ce nœud ;
- (lignes 12 :16) Si les variables u_n sont binaires dans la relaxation SDP (à une précision donnée), une borne supérieure est calculée en utilisant notre fonction `solve_MINLP` selon le dictionnaire de variables fixées. Si ce calcul améliore la borne supérieure, nous la mettons à jour et nous testons si certains nœuds peuvent être coupés (si la borne inférieure de leur nœud-père est strictement supérieure à la nouvelle borne supérieure) ;
- (lignes 17 :24) Sinon, deux nœuds-fils sont créés en branchant sur une variable u_i. Nous choisissons la variable u_i qui est la plus proche de 1 dans la solution SDP. Les deux nœuds-fils ont leur borne inférieure père égale à la borne inférieure calculée avec la relaxation SDP ligne 9. Leur dictionnaire de variables fixées diffère d'une valeur :

il est égal au dictionnaire de variables fixées courant avec une entrée de plus, l'une correspondant à $u_i = 1$ et l'autre à $u_i = 0$.

Entrées : Instance, solution de la relaxation SDP (ξ_n^*/V_{nn}^*), un seuil bas l, un seuil haut u
Sorties : UB

```
1  Fixer les variables u_n à 0 si ξ*_n/V*_nn ≤ l et à 1 si ξ*_n/V*_nn ≥ u
2  fixing0 ← dictionnaire de toutes les variables fixées
3  UB ← solve_MINLP(instance, fixing0)
4  nodes_list ← [node(-∞, fixing0)]
5  tant que |nodes| > 0 faire
6  |  Choisir un nœud n ∈ nodes_list selon la stratégie profondeur d'abord
7  |  fixing ← dictionnaire de toutes les variables fixées au nœud n
8  |  Supprimer le nœud n de la liste nodes_list
9  |  LB = solve_SDP(instance, fixing)
10 |  si LB>UB ou SDP non réalisable alors
11 |  |  Nœud coupé
12 |  sinon
13 |  |  si variables u_i binaires dans la solution SDP alors
14 |  |  |  UB' ← solve_MINLP(instance, fixing)
15 |  |  |  UB ← UB' if UB'< UB
16 |  |  |  Supprimer les nœuds de la liste nodes_list pour lesquels LB>UB
17 |  |  |  Nœud coupé
18 |  |  sinon
19 |  |  |  Création de deux nœuds
20 |  |  |  Brancher sur la variable u_i qui est la plus proche de 1 dans la solution SDP
21 |  |  |  Créer deux copies fixing0 et fixing1 du dictionnaire de variables fixées
22 |  |  |  Fixer la variable u_i à 0 dans fixing0
23 |  |  |  Fixer la variable u_i à 1 dans fixing1
24 |  |  |  Ajouter le nœud node(LB, fixing0) à la liste nodes_list
25 |  |  |  Ajouter le nœud node(LB, fixing1) à la liste nodes_list
26 |  |  fin
27 |  fin
28 fin
```

Algorithme 4 : Pseudo-code de notre procédure d'énumération implicite s'appuyant sur une relaxation SDP

Une relaxation SDP est bien plus coûteuse qu'une relaxation linéaire mais c'est une relaxation beaucoup plus précise, surtout pour les problèmes d'OPF. Utiliser une relaxation SDP dans une procédure d'énumération implicite n'est un choix judicieux que si l'arbre d'exploration est de taille raisonnable. C'est pourquoi nous proposons une procédure qui énumère seulement les variables binaires ; nous ne faisons pas d'énumération implicite spatiale (*Branch-and-Bound spatial*) qui serait beaucoup trop coûteuse. Notre méthode n'est donc pas une

méthode exacte : même si toutes les variables binaires sont fixées, il est possible qu'il y ait un saut d'optimalité non nul entre la borne inférieure et la borne supérieure (si la relaxation SDP n'est pas exacte ou si la solution réalisable calculée par un solveur non linéaire local n'est pas un optimum global). Notre objectif est plutôt de trouver la meilleure solution réalisable à partir de la solution de la relaxation SDP dans une limite de temps raisonnable, ce qui correspond aux attentes de RTE qui préfère calculer de bonnes solutions réalisables rapidement plutôt qu'obtenir la solution optimale à tout prix. Nous pouvons voir notre méthode comme un compromis entre une méthode exacte extrêmement coûteuse et une simple procédure d'arrondi. Pour rendre notre méthode un peu moins coûteuse, nous choisissons de fixer des variables binaires à partir de la solution de la relaxation SDP. Ce choix est d'autant plus pertinent que rien ne garantit que notre méthode améliore la solution. Il est donc prudent de restreindre l'arbre d'exploration. Nous choisissons différentes stratégies de fixation en fonction du problème.

- Pour le problème ROPF avec la contrainte MAXkshunts, beaucoup de MCS ne seront pas mis en service. Nous proposons donc de fixer à 0 toutes les variables qui sont inférieures à 0.25 dans la solution de la relaxation SDP. Ce seuil de 0.25 permet de fixer un certain nombre de variables binaires tout en laissant suffisamment de choix pour améliorer la solution.
- Pour le problème avec la contrainte MAXkmoves, tout dépend de la situation initiale. Dans notre cas, nous calculons une situation initiale en arrondissant une solution de la relaxation continue du problème. Cette situation initiale comporte beaucoup de MCS en service. Nous décidons donc de fixer à 1 les variables qui sont supérieures à 0.75 dans la solution de la relaxation SDP. Ce seuil de 0.75 permet encore une fois un bon compromis entre le nombre de variables binaires fixées et le nombre de variables binaires libres.
- Pour le problème avec la contrainte de production GENmoves, nous avons moins d'information sur les MCS. Nous proposons donc de fixer à 0 les variables qui sont inférieures à 10^{-4} dans la solution de la relaxation SDP et à 1 les variables qui sont supérieures à 0.9. Nous avons choisi les seuils de 10^{-4} et 0.9 en testant l'impact de la fixation sur la borne inférieure. Pour ces valeurs, la borne inférieure n'est pas significativement détériorée.

5.5 Résultats numériques

Dans cette section, nous présentons des résultats numériques pour les trois variantes du problème ROPF. Nous montrons d'abord dans la section 5.5.1 que la résolution de la re-

laxation SDP est accélérée en utilisant un algorithme de fusion de cliques approprié. Nous présentons ensuite dans la section 5.5.2 les performances de notre algorithme 4. Pour les trois variantes, les tests ont été réalisés sur un processeur Intel® Core™ i7-6820HQ CPU @2.70GHz. Nous avons utilisé Julia 1.0.3. [83] pour implémenter notre algorithme d'énumération implicite. Nous avons construit les problèmes ROPF en utilisant notre module MathProgComplex.jl [93]. Les modules JuMP.jl 0.19.0 [84] et Mosek.jl avec MOSEK 9.1 [100] ont été utilisés pour la résolution des problèmes SDP. Le langage de modélisation AMPL Version 2018062 [85] et le solveur Knitro 11.0.1 ont été utilisés pour la résolution locale des MINLPs.

5.5.1 Accélération de la résolution de la relaxation SDP grâce à la fusion de cliques

Le tableau 5.1 présente les résultats de la comparaison entre deux types de décomposition en cliques pour le problème ROPF avec la contrainte optionnelle (MAXkshunts). La première décomposition en cliques est une décomposition classique obtenue en appliquant une factorisation de Cholesky précédée d'un ordre AMD. La deuxième décomposition est obtenue à partir de la première sur laquelle est appliquée notre algorithme de fusion de cliques présenté dans la section 4.4 (avec $k_{max} = 1$). Comme indiqué dans la section 5.2, le paramètre **k** est égal à 4 par défaut car c'est une valeur qui est utilisée chez RTE dans les études. Si cette valeur conduit à un problème non réalisable (que cela soit prouvé ou supposé), nous choisissons **k** comme la plus petite valeur pour laquelle il est possible de calculer une solution réalisable. Nous nous intéressons aux instances MATPOWER de plus de 1000 nœuds. Nous observons que le temps de résolution MOSEK est significativement diminué grâce à l'algorithme de fusion de cliques. De manière grossière, le temps de résolution est divisé par deux. Plus précisément, l'amélioration moyenne est de 129%. Cette amélioration est primordiale pour un algorithme d'énumération implicite dans lequel il faut résoudre une relaxation SDP à chaque nœud. Des résultats similaires sont observés sur les deux autres variantes du problème ROPF.

5.5.2 Performances de l'algorithme 4

Nous comparons notre algorithme à la seule alternative présente dans la littérature pour résoudre le problème ROPF à l'aide d'une relaxation convexe : une procédure d'arrondi. Cette procédure d'arrondi est la suivante : nous fixons à 1 toutes les variables supérieures ou égales à 0.5 (les **k** plus grandes s'il y en a plus que **k** pour le problème avec la contrainte MAXkshunts), le reste à 0 et nous utilisons le solveur local non linéaire Knitro pour calculer une solution réalisable. Les tableaux 5.2, 5.3, 5.4 et 5.5 présentent les résultats de cette

Tableau 5.1 Temps de résolution MOSEK pour les instances MATPOWER de plus de 1000 nœuds

Instance	k	Temps de résolution MOSEK (secondes)	
		Pas de fusion	Fusion de cliques
case1354pegase	4	12.28	6.72
case1888rte	4	11.50	7.09
case1951rte	4	11.50	8.41
case_ACTIVSg2000	4	475.76	86.45
case2383wp	4	93.61	29.36
case2736sp	4	105.77	49.55
case2737sop	4	119.11	47.34
case2746wop	4	115.67	66.80
case2746wp	4	96.44	53.11
case2848rte	4	29.58	11.69
case2868rte	4	21.88	11.16
case2869pegase	12	48.17	26.05
case3012wp	4	151.55	62.19
case3120sp	4	167.83	73.19
case3375wp	4	120.36	60.52
case6468rte	4	259.48	111.08
case6470rte	4	243.70	113.14
case6495rte	14	226.61	111.77
case6515rte	66	199.13	95.11
case9241pegase	125	922.20	388.76
case13659pegase	1100	269.56	123.63

comparaison. Ils ont tous la même structure. Les premières colonnes donnent des informations à propos de l'instance : son nom, le nombre de MCS $|S|$ et éventuellement une indication à propos de la variante du problème. Les trois colonnes suivantes détaillent l'encadrement de la valeur optimale obtenu avec la procédure présentée dans la section 5.4.1 (borne supérieure UB, borne inférieure LB et saut d'optimalité $\frac{\text{UB-LB}}{\text{UB}}$ donné en pourcentage). Les 5 colonnes suivantes montrent les résultats de notre méthode d'énumération implicite pour les instances pour lesquelles le saut d'optimalité n'est pas satisfaisant, i.e., strictement supérieur à 10^{-4}. La première colonne de cette partie présente le nombre de variables binaires qui sont libres dans notre procédure d'énumération implicite. Les deux suivantes donnent le nombre de nœuds parcourus et le temps en secondes de la procédure d'énumération implicite. Nous utilisons une limite de temps de 3600 secondes. Quand cette limite de temps implique un arrêt prématuré de la procédure, nous notons '>3600'. La colonne suivante présente la meilleure borne supérieure calculée par notre algorithme : avec la procédure d'énumération implicite dans le temps limite imposé ou avec notre heuristique en trois étapes si la borne calculée par l'algorithme d'énumération implicite est moins bonne, ce qui peut arriver puisque nous fixons des variables binaires. La colonne suivante montre le saut d'optimalité obtenu avec cette (nouvelle) borne supérieure. Les deux dernières colonnes montrent les résultats de la procédure d'arrondi : la borne supérieure calculée et le saut d'optimalité correspondant.

ROPF avec contrainte MAXkshunts

Nous appliquons d'abord notre méthode sur le problème ROPF avec la contrainte optionnelle (MAXkshunts). Les résultats de la comparaison avec une méthode d'arrondi sont présentés dans le tableau 5.2. La troisième colonne indique la valeur de **k** utilisée. Nous observons que 13 instances sur 31 sont résolues durant la première étape de notre procédure et 7 de plus grâce à l'énumération implicite, soit 20 instances au total contre 14 pour la procédure d'arrondi. De plus, notre procédure est plus robuste que la procédure d'arrondi : il y a 9 instances pour lesquelles la procédure d'arrondi ne donne pas de solution réalisable versus 1 seule pour notre procédure (case6470rte). Cependant, si l'on double le temps limite, une solution réalisable est calculée pour l'instance case6470rte (UB=98697.64, gap=0.05%) et cela en exactement 5382 secondes. Il est aussi intéressant de noter que la procédure d'arrondi est généralement inefficace pour les grands réseaux (plus de 6000 nœuds) alors que notre algorithme calcule une solution réalisable dès le début (sauf pour l'instance case6470rte). Les solutions réalisables calculées sont plutôt de bonne qualité : le saut d'optimalité obtenu à la fin de notre procédure pour les deux plus gros réseaux est de l'ordre de 1% mais est inférieur ou égal à 0.52% pour les autres instances. Au niveau du temps de calcul, nous remarquons que les instances qui sont résolues le sont en moins de 500 secondes. Pour les

autres instances (excepté case_ACTIVSg500), l'énumération implicite n'est pas finie en 3600 secondes. Toutefois, si le temps limite est suffisamment augmenté, l'énumération implicite se termine pour certaines instances : case1888rte en 19770 secondes, case_ACTIVSg2000 en 7249 secondes, case2848rte en 6747 secondes, case3120sp en 11450 secondes mais il n'y a pas d'amélioration de la borne supérieure. Pour les grands réseaux de plus de 6000 nœuds, l'énumération implicite prend au moins huit heures et la borne supérieure n'est pas toujours améliorée après ces huit heures de calculs.

Tableau 5.2 Résultats pour le problème ROPF avec la contrainte (MAXkshunts)

Instance	$\|S\|$	k	Encadrement de la valeur optimale			Énumération implicite					Arrondi	
			UB	LB	Saut	#binvar	#nœuds	Temps (s)	UB	Saut	UB	Saut
14	1	4	5371.50	5371.50	**0.00%**						5371.50	**0.00%**
24_ieee_rts	1	4	44259.19	44259.19	**0.00%**						44259.24	**0.00%**
30	2	4	373.41	373.39	**0.00%**						373.41	**0.00%**
ieee30	2	4	5927.59	5927.59	**0.00%**						5927.64	**0.00%**
57	3	4	25337.79	25337.79	**0.00%**						25337.70	**0.00%**
89pegase	44	4	5813.41	5812.94	**0.00%**						5812.96	**0.00%**
118	14	4	86301.50	86298.49	**0.00%**						86301.52	**0.00%**
ACTIVSg200	4	4	34641.24	34640.46	**0.00%**						34640.11	**0.00%**
illinois200	4	4	43763.98	43763.92	**0.00%**						43763.98	**0.00%**
300	29	4	503727.37	475470.69	5.61%	7	13	32.51	475482.55	**0.00%**	475526.23	0.01%
300mod	29	4	476470.28	475475.06	0.21%	7	13	23.28	475482.47	**0.00%**	503035.65	5.48%
ACTIVSg500	15	4	91446.78	90967.34	0.52%	5	47	102.17	91446.78	0.52%	91454.53	0.53%
1354pegase	1082	4	Inf	74104.05	-	5	1	18.80	74107.29	**0.00%**	74114.91	0.01%
1888rte	45	4	Inf	59621.43	-	11	181	>3600	59882.62	0.44%	59986.33	0.61%
1951rte	24	4	81838.43	81744.99	0.11%	6	5	70.41	81751.01	**0.00%**	81838.41	0.11%
ACTIVSg2000	149	4	1245341.82	1244050.72	0.10%	7	23	>3600	1245353.63	0.10%	Inf	-
2736sp	1	4	1320569.85	1320559.84	**0.00%**						1320569.91	**0.00%**
2737sop	5	4	793976.68	793733.53	0.03%	5	5	244.95	793749.57	**0.00%**	793749.57	**0.00%**
2746wop	6	4	1223001.54	1222947.85	**0.00%**						1223001.58	**0.00%**
2848rte	48	4	53052.04	53030.69	0.04%	7	106	>3600	53051.60	0.04%	Inf	-
2868rte	33	4	79836.41	79832.52	**0.00%**						Inf	-
2869pegase	2197	12	Inf	134155.54	-	14	13	434.81	134155.82	**0.00%**	134185.65	0.02%
3012wp	9	4	2582246.71	2581933.31	0.01%	5	5	457.80	2582191.41	**0.00%**	2582191.41	**0.00%**
3120sp	9	4	2141382.41	2139564.66	0.08%	7	38	>3600	2141382.41	0.08%	Inf	-
3375wp	9	4	7404662.46	7404199.17	**0.00%**						7404269.18	**0.00%**
6468rte	97	4	86938.67	86869.63	0.08%	8	23	>3600	86938.67	0.08%	Inf	-
6470rte	73	4	Inf	98650.43	-	6	23	>3600	Inf	-	Inf	-
6495rte	99	14	106740.51	106310.40	0.40%	17	26	>3600	106740.51	0.40%	Inf	-
6515rte	102	66	110238.10	109913.67	0.29%	74	26	>3600	110238.10	0.29%	Inf	-
9241pegase	7327	125	315980.89	311438.46	1.44%	165	18	>3600	315980.89	1.44%	Inf	-
13659pegase	8754	1100	385680.71	380743.60	1.28%	1237	28	>3600	385680.71	1.28%	385675.83	1.28%

ROPF avec contrainte MAXkmoves

Nous testons maintenant notre approche sur le problème ROPF avec la contrainte optionnelle (MAXkmoves). Comme indiqué dans la section 5.2, nous utilisons $\mathbf{k} = 4$ car c'est une valeur utilisée chez RTE dans les études. Nous calculons un état initial des MCS $\mathbf{u^0}$ en arrondissant la solution de la relaxation continue du problème (ROPF) sans contrainte optionnelle. Nous aurions pu choisir un état initial aléatoire mais nous préférions utiliser un état initial plus réaliste.

Les résultats de la comparaison avec une méthode d'arrondi sont présentés dans le tableau 5.3.

Tableau 5.3 Résultats pour le problème ROPF avec la contrainte (MAXkmoves) pour $k = 4$

Instance	$\|S\|$	Encadrement de la valeur optimale			Énumération implicite					Arrondi	
		UB	LB	Saut	#binvar	#nœuds	Temps (s)	UB	Saut	UB	Saut
case14	1	5371.50	5371.50	**0.00%**						5371.50	**0.00%**
24_ieee_rts	1	44259.19	44259.19	**0.00%**						44259.24	**0.00%**
30	2	373.41	373.39	**0.00%**						373.41	**0.00%**
ieee30	2	5927.59	5927.59	**0.00%**						5927.64	**0.00%**
57	3	25337.79	25337.79	**0.00%**						25337.70	**0.00%**
89pegase	44	5812.69	5812.69	**0.00%**						5812.69	**0.00%**
118	14	86298.82	86296.28	**0.00%**						86299.64	**0.00%**
ACTIVSg200	4	34641.15	34640.46	**0.00%**						34640.11	**0.00%**
illinois200	4	43763.98	43763.92	**0.00%**						43763.98	**0.00%**
300	29	475394.34	475379.12	**0.00%**						475395.09	**0.00%**
300mod	29	475396.68	475382.09	**0.00%**						475395.68	**0.00%**
ACTIVSg500	15	91439.37	90782.00	0.72%	0	1	17.38	91343.17	0.61%	91343.17	0.61%
1354pegase	1082	74049.71	74045.79	**0.00%**						74049.81	**0.00%**
1888rte	45	59840.90	59610.98	0.38%	10	336	>3600	59825.10	0.36%	59841.60	0.39%
1951rte	24	81746.93	81736.90	0.01%	5	7	70.36	81742.62	**0.00%**	81748.30	0.01%
ACTIVSg2000	149	1242319.11	1241812.60	0.04%	54	17	1512.15	1242134.91	0.03%	1242202.90	0.03%
2736sp	1	1320569.85	1320553.46	**0.00%**						1320569.91	**0.00%**
2737sop	5	793976.68	793735.65	0.03%	1	1	57.35	793749.61	**0.00%**	793746.84	**0.00%**
2746wop	6	1223001.54	1222994.89	**0.00%**						1223001.58	**0.00%**
2848rte	48	53043.10	53027.08	0.03%	27	62	>3600	53042.10	0.03%	Infeasible	-
2868rte	33	79829.59	79827.34	**0.00%**						Inf	-
2869pegase	2197	133953.39	133948.52	**0.00%**						133953.35	**0.00%**
3012wp	9	2582185.13	2581915.43	**0.00%**						Infeasible	-
3120sp	9	Inf	2139522.22	-	2	5	829.46	2141455.77	0.09%	Inf	-
3375wp	9	7404267.03	7404180.26	**0.00%**						Inf	-
6468rte	97	Inf	86863.18	-	41	20	>3600	86889.13	0.03%	Inf	-
6470rte	73	Inf	98633.73	-	26	14	3417.24	98642.4	**0.00%**	Inf	-
6495rte	99	Inf	106303.77	-	42	15	>3600	106724.29	0.39%	Inf	-
6515rte	102	110253.47	109921.01	0.30%	30	18	>3600	110250.40	0.30%	Inf	-
9241pegase	7327	315674.22	311071.20	1.46%	1549	12	>3600	315674.22	1.46%	Inf	-
13659pegase	8754	Inf	380668.34	-	2283	10	>3600	385572.62	1.27%	Inf	-

Nous observons que 18 instances sont résolues pendant la première étape de notre procédure et 3 de plus grâce à l'énumération implicite, soit un total de 21 instances sur 31. La procédure d'arrondi n'en résout que 16. Encore une fois, notre méthode est plus robuste puisqu'elle calcule une solution réalisable pour toutes les instances, ce qui n'est pas le cas de la procédure d'arrondi. Il y a même 2 instances qui sont non réalisables avec la fixation des variables binaires produite par la procédure d'arrondi (case2848rte et case3012wp). Finalement, même si le saut d'optimalité n'est pas satisfaisant pour toutes les instances, notre procédure donne des sauts raisonnables : entre 1.25% et 1.5% pour les deux plus gros réseaux et moins d'1% pour les autres instances. Cependant, il y a trois instances pour lesquelles notre procédure d'énumération implicite n'améliore pas significativement la solution (case2848rte, case6515rte et case9241pegase). Il y a plusieurs explications possibles : cela peut être dû à la limite de temps, à la non convexité du problème ou encore à la fixation des variables binaires (e.g. une fixation trop restrictive). C'est un risque inhérent à notre procédure d'énumération implicite non exacte. C'est pourquoi nous sommes convaincus qu'il faut imposer une limite de temps pour éviter de perdre du temps à ne rien améliorer. Pour résumer, notre procédure est plus robuste qu'une procédure d'arrondi et elle permet au mieux de clore le saut d'optimalité, au pire d'obtenir une bonne solution réalisable.

ROPF avec contrainte GENmoves

Nous testons finalement notre procédure sur le problème ROPF avec la contrainte optionnelle GENmoves. Pour ces derniers tests, nous supprimons les limites thermiques sur les lignes (borne supérieure sur le module de l'intensité) car les contraintes réelles sur les lignes ne sont pas aussi strictes : il existe plusieurs limites qu'il est possible de franchir à condition de ne pas les dépasser plus d'un certain temps. Nous considérons ici que ces limites peuvent être gérées en post-traitement (ce qui est le cas pour la plupart des simulations chez RTE). Nous calculons plusieurs plans de génération active $\mathbf{P^0}$ en utilisant des valeurs de productions aléatoires. Plus précisément, nous tirons aléatoirement une valeur entre $\mathbf{P_n^{min}}$ et $\mathbf{P_n^{max}}$ pour chaque moyen de production $n \in G$. Ensuite nous vérifions si le total de production active est supérieur ou égal à 1.02 fois la somme des consommations, ce qui correspond à la consommation totale plus une estimation des pertes habituelles. Si cette contrainte est satisfaite, nous avons terminé. Sinon, nous ré-allouons proportionnellement la production manquante. Toutefois, à cause de la borne supérieure $\mathbf{P_n^{max}}$, cette redistribution peut ne pas être suffisante. Dans ce cas, nous parcourons aléatoirement la liste des générateurs et nous augmentons ceux qui ont encore de la marge jusqu'à ce que la contrainte de production soit satisfaite. Nous testons cinq plans de production pour chaque instance. Nous nous concentrons sur les instances de plus de 1000 nœuds et nous écartons les instances pour lesquelles nous ne sommes pas certains qu'une

solution réalisable existe.

Pour éviter les variables binaires δ^- et δ^+, nous résolvons deux problèmes pour chaque instance : l'un avec la variable λ^- (i.e. $\delta^- = 1$ et $\delta^+ = 0$) ce qui correspond à une baisse de production et l'autre avec la variable λ^+ (i.e. $\delta^- = 0$ et $\delta^+ = 1$) ce qui correspond à une augmentation de production.

Les résultats de la comparaison avec une méthode d'arrondi sont présentés dans le tableau 5.4 pour les instances MATPOWER qui comportent entre 1000 et 2850 nœuds. Chaque ligne du tableau correspond à un couple (instance, plan de production active). Il y a 7 instances et 5 plans de production active pour chaque instance, soit 35 cas tests au total. La colonne '+/-' indique si la solution consiste à augmenter ou diminuer le plan de production. Notre procédure résout 11 cas tests lors de la première étape et 9 de plus avec l'énumération implicite. La procédure d'arrondi résout autant de cas tests que la nôtre, c'est-à-dire 20 cas tests sur 35. Notre procédure est plus robuste puisqu'elle calcule toujours une solution réalisable, qui plus est de bonne qualité (saut d'optimalité inférieur à 0.1%), contrairement à la procédure d'arrondi pour laquelle 9 cas tests n'ont pas de solution. En revanche, notre procédure d'énumération n'améliore pas toujours la borne supérieure. Cela peut être dû à la limite en temps qui est assez restrictive. Cette limite est toutefois nécessaire puisque notre procédure n'est pas une méthode exacte et qu'il n'y a aucune garantie d'amélioration de la solution.

Le tableau 5.5 présente les résultats pour les instances MATPOWER de plus de 2850 nœuds (et moins de 9000 nœuds). Il y a 8 instances et 5 plans de production pour chaque instance soit 40 cas tests au total. Pour ces réseaux plus grands, la procédure d'arrondi résout 8 cas tests contre 6 pour la nôtre (2 pendant la première étape et 4 de plus avec l'énumération implicite). Cependant, notre algorithme est toujours plus robuste puisqu'il calcule une solution réalisable pour chaque cas test, alors que la procédure d'arrondi ne donne pas de solution pour 21 cas tests. Plus particulièrement, la procédure d'arrondi est inefficace sur les gros réseaux (plus de 6000 nœuds) tandis que notre procédure calcule des solutions réalisables de très bonne qualité, i.e. avec un saut d'optimalité en-dessous de 0.1%. Cependant, notre procédure d'énumération implicite n'améliore pas la solution (ou pas significativement) pour 34 cas tests sur 38, ce qui montre les limites de notre méthode.

5.6 Conclusion et perspectives

Il y a quatre contributions principales dans ce chapitre. Premièrement, nous avons introduit de nouveaux aspects dans le problème de l'OPF réactif afin de mieux répondre aux besoins

Tableau 5.4 Résultats pour le problème ROPF avec la contrainte GENmoves pour les instances MATPOWER de moins de 2850 nœuds

Instance	$\|S\|$	+/-	Encadrement de la valeur optimale			Énumération implicite					Arrondi	
			UB	LB	Saut	#binvar	# nœuds	Temps (s)	UB	Saut	UB	Saut
1354pegase	1082	-	74454.57	74446.65	0.01%	301		>3600	74454.57	0.01%	74450.93	**0.00%**
1354pegase	1082	+	74560.06	74550.55	0.01%	270	871	>3600	74560.06	0.01%	74556.06	**0.00%**
1354pegase	1082	+	74668.53	74654.90	0.02%	283	883	>3600	74665.98	0.01%	74661.81	**0.00%**
1354pegase	1082	-	74594.60	74583.01	0.02%	329	805	>3600	74594.60	0.02%	74587.74	**0.00%**
1354pegase	1082	+	74603.54	74592.73	0.01%	327	906	>3600	74603.54	0.01%	74597.49	**0.00%**
1951rte	24	-	Inf	82091.31	-	5	21	296.10	82095.94	**0.00%**	Inf	-
1951rte	24	-	82204.12	82194.85	0.01%	5	7	84.77	82198.36	**0.00%**	Inf	-
1951rte	24	-	82234.35	82229.82	**0.00%**						82240.59	0.01%
1951rte	24	-	Inf	82060.86	-	7	19	289.74	82066.48	**0.00%**	Inf	-
1951rte	24	-	Inf	82176.29	-	5	24	409.15	82183.64	**0.00%**	Inf	-
ACTIVSg2000	149	+	68554.86	68514.50	0.06%	109	53	>3600	68554.86	0.06%	68554.38	0.06%
ACTIVSg2000	149	+	68525.15	68489.58	0.05%	115	47	>3600	68525.15	0.05%	68525.69	0.05%
ACTIVSg2000	149	-	68443.64	68407.81	0.05%	114	52	>3600	68443.64	0.05%	68444.34	0.05%
ACTIVSg2000	149	-	68438.09	68400.88	0.05%	113	54	>3600	68438.09	0.05%	68438.90	0.06%
ACTIVSg2000	149	+	68463.58	68451.39	0.02%	112	52	>3600	68463.58	0.02%	68465.25	0.02%
2736sp	1	-	18363.68	18363.44	**0.00%**						18363.68	**0.00%**
2736sp	1	-	18369.37	18369.26	**0.00%**						18369.37	**0.00%**
2736sp	1	-	18378.56	18378.36	**0.00%**						18378.56	**0.00%**
2736sp	1	-	18366.73	18366.45	**0.00%**						18366.73	**0.00%**
2736sp	1	-	18371.55	18371.36	**0.00%**						18371.55	**0.00%**
2737sop	5	-	11418.83	11416.91	0.02%	1	2	78.76	11417.15	**0.00%**	11417.12	**0.00%**
2737sop	5	-	11419.29	11416.84	0.02%	1	2	64.42	11417.07	**0.00%**	11416.99	**0.00%**
2737sop	5	-	11413.85	11411.50	0.02%	1	2	62.63	11411.74	**0.00%**	11411.67	**0.00%**
2737sop	5	-	11417.89	11415.66	0.02%	1	2	70.24	11415.92	**0.00%**	11415.84	**0.00%**
2737sop	5	-	11416.65	11414.10	0.02%	1	2	79.26	11414.38	**0.00%**	11414.30	**0.00%**
2746wop	6	-	19273.40	19273.16	**0.00%**						19273.40	**0.00%**
2746wop	6	-	19263.41	19263.15	**0.00%**						19263.41	**0.00%**
2746wop	6	-	19289.75	19289.39	**0.00%**						19289.59	**0.00%**
2746wop	6	-	19286.57	19286.04	**0.00%**						19286.57	**0.00%**
2746wop	6	-	19285.37	19284.97	**0.00%**						19285.25	**0.00%**
2848rte	48	-	Inf	53142.85	-	34	140	>3600	53163.39	0.04%	Inf	-
2848rte	48	-	Inf	53185.72	-	31	130	>3600	53206.27	0.04%	Inf	-
2848rte	48	-	Inf	53170.47	-	33	132	>3600	53193.82	0.04%	Inf	-
2848rte	48	-	Inf	53191.46		29	149	>3600	53210.87	0.04%	Inf	-
2848rte	48	-	53201.04	53175.97	0.05%	30	135	>3600	53201.04	0.05%	Inf	-

Tableau 5.5 Résultats pour le problème ROPF avec la contrainte GENmoves pour les instances de plus de 2850 nœuds

Instance	$\|S\|$	+/-	Encadrement de la valeur optimale			Énumération implicite					Arrondi	
			UB	LB	Saut	#binvar	#nœuds	Temps (s)	UB	Saut	UB	Saut
2868rte	33	-	80021.98	80016.75	**0.00%**						Inf	-
2868rte	33	-	Inf	80100.29	-	8	10	261.76	80101.64	**0.00%**	80101.64	**0.00%**
2868rte	33	-	Inf	80113.82	-	9	11	2652.89	80116.18	**0.00%**	80115.97	**0.00%**
2868rte	33	-	Inf	80098.29	-	9	11	256.93	80100.42	**0.00%**	80100.42	**0.00%**
2868rte	33	-	Inf	80155.48	-	9	28	735.94	80158.09	**0.00%**	Inf	-
2869pegase	2197	-	134844.35	134829.63	0.01%	577	307	>3600	134844.35	0.01%	134836.89	**0.00%**
2869pegase	2197	+	135468.07	135451.33	0.01%	438	345	>3600	135468.0721	0.01%	135459.81	**0.00%**
2869pegase	2197	+	135221.95	135193.34	0.02%	471	170	>3600	135221.95	0.02%	135202.13	**0.00%**
2869pegase	2197	-	134999.53	134976.67	0.02%	493	308	>3600	134999.53	0.02%	134985.34	**0.00%**
2869pegase	2197	+	135378.32	135357.60	0.02%	511	345	>3600	135378.32	0.02%	135366.56	**0.00%**
3012wp	9	+	27771.36	27755.06	0.06%	2	7	285.85	27770.66	0.06%	27770.66	0.06%
3012wp	9	+	27844.23	27831.10	0.05%	1	3	109.16	27844.10	0.05%	27844.10	0.05%
3012wp	9	+	27810.15	27795.77	0.05%	1	3	116.11	27809.83	0.05%	27809.83	0.05%
3012wp	9	+	27767.51	27752.24	0.05%	1	3	156.41	27767.26	0.05%	27767.28	0.05%
3012wp	9	+	27799.43	27784.16	0.05%	1	3	125.86	27798.93	0.05%	27798.93	0.05%
3120sp	9	+	21642.54	21627.63	0.07%	2	7	326.77	21642.43	0.07%	21642.43	0.07%
3120sp	9	+	21779.92	21765.93	0.06%	1	3	124.55	21779.87	0.06%	21779.87	0.06%
3120sp	9	+	21691.65	21677.24	0.07%	2	7	308.49	21691.65	0.07%	21691.67	0.07%
3120sp	9	+	21678.85	21663.55	0.07%	2	7	258.10	21678.85	0.07%	21678.93	0.07%
3120sp	9	+	21635.82	21620.20	0.07%	2	5	238.77	21635.24	0.07%	21635.24	0.07%
6468rte	97	+	87158.54	87105.29	0.06%	64	55	>3600	87158.54	0.06%	Inf	-
6468rte	97	+	87044.03	87022.41	0.02%	71	48	>3600	87044.03	0.02%	Inf	-
6468rte	97	+	87138.68	87086.74	0.06%	68	51	>3600	87138.68	0.06%	Inf	-
6468rte	97	+	87215.35	87215.35	**0.00%**						Inf	-
6468rte	97	+	87285.51	87246.14	0.05%	65	45	>3600	87285.51	0.05%	Inf	-
6470rte	73	+	98745.81	98733.04	0.01%	36	50	>3600	98745.81	0.01%	Inf	-
6470rte	73	+	98831.51	98796.99	0.03%	47	50	>3600	98831.51	0.03%	Inf	-
6470rte	73	+	98706.23	98683.05	0.02%	47	51	>3600	98706.23454	0.02%	Inf	-
6470rte	73	+	98698.39	98677.03	0.02%	43	48	>3600	98698.39249	0.02%	Inf	-
6470rte	73	+	98758.59	98748.08	0.01%	42	50	>3600	98758.59	0.01%	Inf	-
6495rte	99	-	106406.16	106383.20	0.02%	47	30	>3600	106406.16	0.02%	Inf	-
6495rte	99	+	106318.61	106302.86	0.01%	44	37	>3600	106318.61	0.01%	Inf	-
6495rte	99	+	106585.90	106573.25	0.01%	40	52	>3600	106585.9033	0.01%	Inf	-
6495rte	99	+	106519.60	106506.92	0.01%	46	36	>3600	106519.5953	0.01%	Inf	-
6495rte	99	+	106556.86	106535.65	0.02%	43	36	>3600	106556.86	0.02%	106551.00	0.01%
6515rte	102	+	110106.56	110084.21	0.02%	33	35	>3600	110106.56	0.02%	Inf	-
6515rte	102	+	110034.86	110008.20	0.02%	32	37	>3600	110034.86	0.02%	Inf	-
6515rte	102	+	110066.07	110045.04	0.02%	32	40	>3600	110066.07	0.02%	Inf	-
6515rte	102	+	110114.50	110092.21	0.02%	33	42	>3600	110114.5045	0.02%	Inf	-
6515rte	102	+	110128.23	110102.90	0.02%	31	43	>3600	110128.23	0.02%	Inf	-

de RTE. Plus précisément, nous nous sommes focalisés sur les MCS et nous avons ajouté de nouvelles contraintes dans le problème (limitation du nombre de MCS qui peuvent être mis en service ou qui peuvent être différents d'une situation initiale, contrainte sur la production active pour simuler le réglage primaire de fréquence). Deuxièmement, nous avons proposé une méthode d'énumération implicite flexible, s'appuyant sur une relaxation SDP, pour essayer de résoudre globalement ces problèmes. A notre connaissance, c'est la première fois qu'une méthode d'énumération implicite est testée pour le problème ROPF. Nous avons montré sur les instances MATPOWER que notre méthode permettait de calculer la solution optimale globale dans la plupart des cas. Dans tous les cas, elle permet de calculer une bonne solution réalisable dans une limite de temps raisonnable. Troisièmement, nous avons montré sur les instances MATPOWER que notre algorithme était capable de calculer de meilleures solutions qu'une méthode d'arrondi. Il est aussi plus robuste qu'une méthode d'arrondi. Quatrièmement, nous avons calculé de bonnes solutions pour de gros réseaux (à partir de 6000 nœuds et jusqu'à plus de 10000 nœuds) alors qu'il semble que la plus grande instance considérée dans la littérature à ce jour contienne un peu plus de 3000 nœuds (dans [118]).

Ce travail a été soumis dans le journal JOGO (Journal of Global Optimization) [128].

Les principales perspectives de recherche consistent à résoudre des versions complexifiées des problèmes ROPF présentés dans ce chapitre, par exemple en incluant d'autres moyens de contrôle de la tension (comme le réglage des ratios de transformation des transformateurs ou les parades topologiques). Une autre perspective de recherche consisterait à étudier l'influence du choix du paramètre **k** afin de réfléchir à une doctrine permettant de choisir une bonne valeur pour ce paramètre.

CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS

Le problème de l'OPF (Optimisation des Flux de Puissance) est un problème d'optimisation dans les réseaux de transport d'électricité dont la résolution globale est aujourd'hui encore un défi. Progresser dans la résolution globale de ce problème représente un enjeu majeur pour RTE (Réseau de Transport d'Electricité) puisque le problème de l'OPF est souvent utilisé comme sous-problème dans des chaînes de simulation du système électrique. Récemment, des méthodes d'optimisation globale prometteuses ont été proposées. La plupart de ces méthodes reposent sur des relaxations SDP (Programmation Semi-Définie) de l'OPF. Néanmoins, la résolution de problèmes SDP de grande taille est très coûteuse, ce qui limite la portée de ces méthodes d'optimisation globale. L'objectif principal de ce projet de recherche était d'accélérer la résolution des relaxations SDP de l'OPF afin de montrer que l'optimisation SDP reste un outil puissant pour l'OPF et ce, même pour des réseaux de grande taille. Ce dernier chapitre présente une synthèse des travaux réalisés en vue d'atteindre cet objectif, les limites des résultats obtenus et les perspectives d'amélioration.

6.1 Synthèse des travaux

Nous avons d'abord conçu et implémenté de nouveaux outils en langage Julia afin de faciliter l'implémentation des problèmes d'OPF et de leurs variantes. Nous avons développé un module générique pour l'optimisation polynomiale en variables complexes et un module flexible pour les problèmes d'OPF. Le premier module permet d'implémenter entièrement des problèmes en variables complexes et de les convertir en variables réelles automatiquement. Le deuxième module répond à un besoin de flexibilité pour la modélisation des problèmes de l'OPF. En effet, dans un contexte de transition énergétique où de nouvelles problématiques apparaissent, il est important de pouvoir traiter facilement de nouvelles variantes du problème de l'OPF. Ces deux outils permettent notamment d'implémenter simplement et rapidement des problèmes d'OPF compliqués comme le problème PSCOPF (OPF Préventif avec Contraintes de Sécurité) qui anticipe des défauts sur le réseau.

Nous avons ensuite étudié les performances de résolution des relaxations SDP de l'OPF en nous concentrant sur les techniques de décomposition en cliques maximales. Nous avons d'abord montré que le choix de la décomposition en cliques maximales a un impact significatif sur le temps de résolution de la relaxation du rang en étudiant différents aspects. Nous avons notamment montré que le choix de l'algorithme d'extension cordale peut avoir un impact significatif sur la résolution en comparant deux algorithmes très classiques : l'heuristique

Degré Minimal et une factorisation de Cholesky précédée d'un ordre AMD (Approximation du Degré Minimal). Nous avons aussi comparé les performances d'une décomposition issue du problème en variables complexes ou du problème en variables réelles. Nous avons démontré sur un petit exemple que l'extension cordale sur le problème réel peut contenir moins d'arêtes que celle issue du problème complexe. Cependant, les tests numériques ont suggéré qu'il était plus pertinent de travailler en variables complexes et ce, même si l'extension cordale contient effectivement plus d'arêtes que dans le cas réel. Nous avons aussi étudié l'impact de la fusion de cliques, qui revient à ajouter des arêtes dans l'extension cordale, en testant différents algorithmes. Nous avons montré que la fusion de cliques peut améliorer significativement les performances de résolution, ce qui confirme qu'ajouter des arêtes dans l'extension cordale peut s'avérer judicieux. Par ailleurs, nous avons proposé une nouvelle stratégie de fusion de cliques pour accélérer la résolution de la relaxation du rang de l'OPF. Nous avons montré que notre algorithme permet d'améliorer significativement les performances de résolution obtenues avec la meilleure heuristique de fusion de cliques pour l'OPF [1] pour les instances MATPOWER. Finalement, nous avons essayé d'adapter ces travaux pour la résolution de la relaxation SDP d'ordre 2 de la hiérarchie de Lasserre mais la structure du problème est trop différente. De plus, nous avons tenté d'implémenter une approche génération de colonnes et/ou contraintes pour résoudre la relaxation d'ordre 2 mais nos tests n'ont pas été concluants.

Nous avons finalement tiré parti de notre algorithme qui accélère la résolution de la relaxation du rang de l'OPF pour résoudre des problèmes d'OPF Réactif (ROPF) via des techniques de programmation semi-définie positive. Nous avons d'abord défini de nouvelles versions du problème ROPF en introduisant trois nouvelles contraintes optionnelles afin de mieux répondre aux besoins de RTE. Deux de ces contraintes se concentrent sur la limitation de la mise en service des MCS (Moyens de Compensation Statique) puisqu'en pratique il n'est pas souhaitable de manipuler constamment ces composants. La troisième contrainte concerne le plan de production active et modélise le réglage primaire de fréquence qui autorise une baisse ou une augmentation coordonnée de production par rapport à un plan de production initial. Pour résoudre ces trois problèmes, nous avons proposé une méthode flexible d'énumération implicite s'appuyant sur une relaxation SDP à chaque nœud. Cette méthode permet de calculer la solution globale ou une très bonne solution réalisable pour les instances MATPOWER dans une limite de temps raisonnable. Nous avons aussi montré que notre méthode était plus performante et plus robuste qu'une méthode d'arrondi, seule méthode utilisée à ce jour pour calculer une solution réalisable du problème de l'OPF réactif à partir d'une relaxation convexe. Pour finir, notre algorithme permet de calculer de bonnes solutions pour de grands réseaux (plus de 6000 nœuds) alors que les plus grands réseaux considérés dans la littérature contiennent environ 3000 nœuds.

6.2 Limitations des solutions proposées

Notre module Julia pour l'optimisation polynomiale en variables complexes ne permet actuellement la conversion en variables réelles qu'en utilisant la représentation rectangulaire pour toutes les variables. Ne pas avoir le choix pour la conversion en variables réelles est la principale limite de notre module. Par ailleurs, notre module pour l'OPF est tellement flexible qu'il peut être difficile à prendre en main : pour ajouter une nouvelle modélisation, l'utilisateur doit définir plusieurs fonctions, ce qui implique de bien comprendre la structure. Notre module pourrait aussi être enrichi des principales relaxations convexes proposées dans la littérature.

La limite la plus évidente concernant nos travaux sur l'accélération de la résolution de la relaxation du rang de l'OPF est qu'ils ne peuvent pas être adaptés facilement pour l'ordre 2 de la hiérarchie de Lasserre car les structures sont trop différentes. Nous n'avons donc pas réussi à accélérer la résolution de la relaxation SDP d'ordre 2 de la hiérarchie de Lasserre avec les méthodes sur lesquelles nous nous sommes concentrés. Par ailleurs, notre stratégie de fusion de cliques pour accélérer la résolution de la relaxation du rang présente aussi des limites. D'abord, elle utilise plusieurs paramètres qui ont été réglés par et pour les instances MATPOWER. Il n'est donc pas garanti que notre algorithme soit performant sur d'autres problèmes. De plus, notre algorithme s'applique en post-traitement sur une décomposition en cliques maximales alors qu'il aurait pu être intéressant de travailler directement sur l'algorithme d'extension cordale. Cependant, travailler directement sur l'algorithme d'extension cordale requiert une compréhension assez fine des caractéristiques d'une bonne décomposition en cliques maximales. Nous n'avons pas clairement identifié de telles caractéristiques. Finalement, notre algorithme a été conçu dans le but de résoudre des problèmes d'OPF différents mais qui gardent la même structure de réseau. Ceci signifie que la décomposition en cliques maximales calculée avec notre algorithme est réutilisée pour plusieurs problèmes. Ainsi il n'est pas problématique de devoir générer plusieurs décompositions afin de régler les paramètres de notre critère pour finalement générer une décomposition en cliques maximales efficace puisque cette décomposition finale est utilisée plusieurs fois. En revanche, notre méthode n'est peut-être pas la meilleure à implémenter pour résoudre un unique problème. Autrement dit, il n'est peut-être pas intéressant de passer du temps à générer une décomposition en cliques maximales efficace si le problème n'est résolu qu'une seule fois.

Nos travaux sur l'OPF réactif comportent plusieurs limites. Tout d'abord, la modélisation pourrait être affinée pour s'approcher un peu plus des problématiques réelles de RTE. Par exemple, la modélisation du réglage primaire de fréquence pourrait être plus précise puisqu'en réalité les groupes de production ne participent pas tous de la même façon et certains arrivent

en butée avant d'autres. Par ailleurs, la minimisation des coûts de production n'est pas toujours une fonction objectif pertinente, par exemple dans des situations court terme dans lesquelles la production est définie à l'avance et n'est pas censée beaucoup varier. Pour finir, notre modélisation ne prend pas en compte tous les moyens de réglage de la tension et pourrait donc être complétée. Ensuite, la méthode d'énumération implicite que nous proposons n'est pas exacte et il n'y a donc aucune garantie de trouver la solution optimale. En pratique, notre méthode d'énumération implicite n'améliore pas toujours la solution réalisable initialement calculée. Par ailleurs, elle dépend fortement du choix de fixation des variables binaires. Notre méthode peut donc être inefficace à cause de la non convexité du problème mais aussi à cause d'une mauvaise fixation des variables binaires. Finalement, nos tests ont été réalisés sur les instances MATPOWER avec des choix de paramètres que nous avons voulu réalistes mais qui restent arbitraires et aucun test n'a été fait sur des données réelles.

6.3 Améliorations futures

La principale amélioration concernant notre module Julia pour l'optimisation polynomiale en variables complexes consisterait à implémenter toutes les conversions en variables réelles possibles afin de pouvoir choisir entre représentation rectangulaire et représentation polaire pour chaque variable. Cependant, l'utilisation de la représentation polaire pour au moins une des variables complexes impliquerait la perte de la structure polynomiale du problème dans la plupart des cas. Il faudrait alors définir de nouvelles structures adaptées à la représentation polaire. La principale amélioration de notre module pour l'OPF consisterait à implémenter les principales relaxations convexes proposées dans la littérature.

La principale piste d'amélioration de nos travaux sur l'accélération de la résolution de la relaxation du rang de l'OPF en utilisant une décomposition en cliques maximales repose sur l'intelligence artificielle. A court terme, nous pensons qu'une méthode d'apprentissage automatique pourrait être utilisée pour définir quelle est la meilleure décomposition en cliques maximales parmi un ensemble de décompositions donné. Cette méthode pourrait mettre en évidence les paramètres d'une bonne décomposition en estimant le temps de résolution en fonction de ces paramètres. Une des difficultés serait toutefois de définir un bon ensemble d'entraînement : suffisamment diversifié mais avec des décompositions en cliques maximales qui restent performantes. A moyen terme, nous pourrions utiliser l'estimation du temps de résolution donnée par une méthode d'apprentissage automatique pour implémenter un nouvel algorithme de fusion de cliques. Si cette estimation nous permet de comprendre clairement les caractéristiques d'une bonne décomposition en cliques, nous pourrions essayer de définir une nouvelle heuristique d'extension cordale qui ne cherche pas à minimiser le nombre

d'arêtes ajoutées dans le graphe mais l'estimation du temps de résolution. Une autre piste d'amélioration consisterait à combiner des techniques de réduction faciale aux techniques de décomposition cordale, comme cela est proposé dans [17]. Nous pourrions ainsi utiliser notre stratégie de fusion de cliques puis utiliser des techniques de réduction faciale. A long terme, la principale perspective que nous envisageons concerne la résolution de la relaxation d'ordre 2 de la hiérarchie de Lasserre. L'ordre 2 conduisant à des problèmes SDP de grande taille, il semblerait judicieux de se tourner vers des méthodes de premier ordre de type ADMM puisque les méthodes de points intérieurs sont beaucoup trop coûteuses pour de telles tailles de problèmes, et ce, même avec les techniques de décomposition en cliques maximales. Les méthodes de premier ordre sont beaucoup moins précises mais peuvent traiter des problèmes assez gros. En particulier, les problèmes d'optimisation à résoudre dans les méthodes de type ADMM (Méthode des Multiplicateurs à Direction Alternée) sont généralement très simples et peu coûteux, ce qui permet un passage à l'échelle. Toutefois, ces méthodes reposent sur l'utilisation du Lagrangien augmenté et il est généralement difficile de trouver une bonne valeur pour le paramètre de pénalisation.

Les pistes d'amélioration concernant nos travaux sur le problème de l'OPF réactif concernent autant la modélisation que la résolution. A court terme, nous envisageons principalement deux pistes de recherche. La première consisterait à tester l'influence du paramètre $\mathbf{k}$ sur la résolution. En effet, puisque nous avons choisi une valeur arbitraire, il serait judicieux de vérifier que notre méthode de résolution reste performante lorsque la valeur de $\mathbf{k}$ varie. La deuxième piste de recherche concerne notre méthode d'énumération implicite que nous pourrions améliorer en testant d'autres stratégies de fixation de variables binaires. La fixation de variables binaires est en effet un bon moyen d'obtenir un compromis entre la qualité de la solution et le temps de résolution. Nous avons proposé certaines stratégies de fixation et nous en avons testé beaucoup d'autres mais il y a encore énormément de possibilités. Il serait aussi intéressant de tester des stratégies d'exploration hybrides (par exemple en commençant par la stratégie profondeur d'abord puis meilleur d'abord). Il nous semble en tout cas peu judicieux de développer une méthode d'énumération implicite spatiale qui serait très coûteuse. A moyen terme, nous envisageons trois axes de recherche. Le premier axe de recherche consisterait à tester notre méthode sur des données réelles afin de confirmer ou d'infirmer les performances obtenues sur les instances MATPOWER. Travailler sur des données réelles nous permettrait aussi de mieux régler le paramètre $\mathbf{k}$. Le deuxième axe de recherche consisterait à tester d'autres fonctions objectif plus adaptées à des problématiques court terme. Par exemple, minimiser la déviation par rapport à un plan de tension défini serait une fonction objectif plus pertinente pour des problématiques de tension à court terme. Le troisième axe de recherche consisterait à renforcer la relaxation SDP utilisée dans l'algorithme

d'énumération implicite afin d'en améliorer les performances. Nous pourrions par exemple tenter d'adapter les relaxations SDP renforcées proposées pour le problème de l'OPF [33] au problème de l'OPF réactif. A plus long terme, nous envisageons deux pistes d'amélioration en terme de modélisation. La première consisterait à prendre en compte d'autres moyens de contrôle (réglage des prises des transformateurs, changements topologiques). La deuxième consisterait à définir une modélisation plus fine du réglage primaire de fréquence. Ces deux options complexifieraient encore le problème (il y aurait notamment beaucoup de variables binaires) et il faudrait probablement définir une nouvelle stratégie de résolution.

De manière plus générale, il serait intéressant de voir si nos travaux peuvent être étendus à un problème d'OPF dans lequel les incertitudes sur la demande ou sur la production (notamment renouvelable) sont prises en compte, via l'optimisation robuste [129], stochastique [130] ou encore avec des méthodes de quantification des risques (*chance constraints*) [131, 132]. Pour commencer, il serait intéressant de regarder si nos résultats (en particulier les temps de calcul) sont sensibles à de petites variations de la demande en électricité ou des capacités de production. Une autre perspective de recherche concerne la réoptimisation qui est une autre manière de gérer les incertitudes. La réoptimisation consiste à résoudre un problème d'optimisation qui possède des similitudes avec un problème que l'on a résolu auparavant en tirant profit de la première résolution pour accélérer la deuxième. En pratique, ceci peut être très utile pour les applications industrielles dans lesquelles il est courant de prendre en compte un état initial. Le problème ROPF avec la contrainte (MAXkmoves) en est un bon exemple : nous optimisons par rapport à une situation initiale donnée concernant les MCS. Ainsi, être capable de réoptimiser un problème d'OPF dans lequel la demande en électricité et/ou les capacités de certains groupes de production auraient changé par rapport à la situation prévue serait un moyen de se prémunir des incertitudes sur ces grandeurs.

RÉFÉRENCES

[1] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre et C. L. DeMarco, "Implementation of a large-scale optimal power flow solver based on semidefinite programming," *IEEE Transactions on Power Systems*, vol. 28, n°. 4, p. 3987–3998, 2013.

[2] J. Carpentier, "Contribution to the economic dispatch problem," *Bulletin de la Societe Francaise des Electriciens*, vol. 3, n°. 8, p. 431–447, 1962.

[3] D. Bienstock et A. Verma, "Strong np-hardness of ac power flows feasibility," *Operations Research Letters*, 2019.

[4] K. Lehmann, A. Grastien et P. Van Hentenryck, "Ac-feasibility on tree networks is np-hard," *IEEE Transactions on Power Systems*, vol. 31, n°. 1, p. 798–801, Jan 2016.

[5] L. Gan, N. Li, U. Topcu et S. H. Low, "Exact convex relaxation of optimal power flow in radial networks," *IEEE Transactions on Automatic Control*, vol. 60, n°. 1, p. 72–87, 2014.

[6] C. Josz, "Application of polynomial optimization to electricity transmission networks," *arXiv preprint arXiv :1608.03871*, 2016.

[7] C. Josz et D. K. Molzahn, "Lasserre hierarchy for large scale polynomial optimization in real and complex variables," *SIAM Journal on Optimization*, vol. 28, n°. 2, p. 1017–1048, 2018.

[8] H. Godard, "Résolution exacte du problème de l'optimisation des flux de puissance," Thèse de doctorat, Paris, CNAM, 2019.

[9] H. Godard, S. Elloumi, A. Lambert, J. Maeght et M. Ruiz, "Novel approach towards global optimality of optimal power flow using quadratic convex optimization," dans *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, April 2019, p. 1227–1232.

[10] A. Gopalakrishnan, A. U. Raghunathan, D. Nikovski et L. T. Biegler, "Global optimization of optimal power flow using a branch & bound algorithm," dans *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2012, p. 609–616.

[11] L. Vandenberghe et S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, n°. 1, p. 49–95, 3 1996.

[12] T. Terlaky, M. F. Anjos et S. Ahmed, *Advances and Trends in Optimization with Engineering Applications*. SIAM, 2017, vol. 24.

[13] Y. Nesterov et A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*. Siam, 1994, vol. 13.

[14] R. Y. Zhang et J. Lavaei, "Modified interior-point method for large-and-sparse low-rank semidefinite programs," dans *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*. IEEE, 2017, p. 5640–5647.

[15] M. Fukuda, M. Kojima, K. Murota et K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion i : General framework," *SIAM Journal on Optimization*, vol. 11, n°. 3, p. 647–674, 2001.

[16] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima et K. Murota, "Exploiting sparsity in semidefinite programming via matrix completion ii : Implementation and numerical results," *Mathematical Programming*, vol. 95, n°. 2, p. 303–327, 2003.

[17] V. Kungurtsev et J. Marecek, "A two-step pre-processing for semidefinite programming," dans *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, p. 384–389.

[18] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart et A. Wynn, "Fast admm for semidefinite programs with chordal sparsity," dans *American Control Conference (ACC), 2017*. IEEE, 2017, p. 3335–3340.

[19] K. K. Sivaramakrishnan, "A parallel interior point decomposition algorithm for block angular semidefinite programs," *Computational Optimization and Applications*, avr. 2008.

[20] J. Mareček et M. Takáč, "A low-rank coordinate-descent algorithm for semidefinite programming relaxations of optimal power flow," *Optimization Methods and Software*, vol. 32, n°. 4, p. 849–871, 2017.

[21] R. D. Zimmerman, C. E. Murillo-Sánchez, R. J. Thomas *et al.*, "Matpower : Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, n°. 1, p. 12–19, 2011.

[22] A. K. Khamees, N. Badra et A. Y. Abdelaziz, "Optimal power flow methods : A comprehensive survey," *International Electrical Engineering Journal (IEEJ)*, vol. 7, n°. 4, p. 2228–2239, 2016.

[23] S. R. Stephen Frank, Ingrida Steponavice, "Optimal power flow : A bibliographic survey i," *Energy Systems*, vol. 3, p. 221–258, 2012.

[24] S. Frank, I. Steponavice et S. Rebennack, "Optimal power flow : a bibliographic survey ii," *Energy Systems*, vol. 3, n°. 3, p. 259–289, 2012.

[25] W. A. Bukhsh, A. Grothey, K. I. McKinnon et P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 28, n°. 4, p. 4780–4788, 2013.

[26] S. H. Low, "Convex relaxation of optimal power flow—part i : Formulations and equivalence," *IEEE Transactions on Control of Network Systems*, vol. 1, n°. 1, p. 15–27, 2014.

[27] ——, "Convex relaxation of optimal power flow—part ii : Exactness," *IEEE Transactions on Control of Network Systems*, vol. 1, n°. 2, p. 177–189, 2014.

[28] D. K. Molzahn et I. A. Hiskens, "A Survey of Relaxations and Approximations of the Power Flow Equations," *Foundations and Trends in Electric Energy Systems*, vol. 4, n°. 1-2, p. 1–221, February 2019.

[29] R. P. O'Neill, A. Castillo et M. B. Cain, "The iv formulation and linear approximations of the ac optimal power flow problem optimal power flow paper 2," *FERC staff technical paper*, déc. 2012.

[30] C. Coffrin et P. Van Hentenryck, "A linear-programming approximation of ac power flows," *INFORMS Journal on Computing*, vol. 26, n°. 4, p. 718–734, 2014.

[31] X. Bai, H. Wei, K. Fujisawa et Y. Wang, "Semidefinite programming for optimal power flow problems," *International Journal of Electrical Power & Energy Systems*, vol. 30, n°. 6-7, p. 383–392, 2008.

[32] J. Lavaei et S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, n°. 1, p. 92–107, févr. 2012.

[33] C. Coffrin, H. L. Hijazi et P. Van Hentenryck, "Strengthening the sdp relaxation of ac power flows with convex envelopes, bound tightening, and valid inequalities," *IEEE Transactions on Power Systems*, vol. 32, n°. 5, p. 3549–3558, 2017.

[34] B. Ghaddar, J. Marecek et M. Mevissen, "Optimal power flow as a polynomial optimization problem," *IEEE Transactions on Power Systems*, vol. 31, n°. 1, p. 539–546, 2015.

[35] C. Bingane, M. F. Anjos et S. Le Digabel, "Tight-and-cheap conic relaxation for the ac optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 33, n°. 6, p. 7181–7188, 2018.

[36] B. Kocuk, S. S. Dey et X. A. Sun, "Strong socp relaxations for the optimal power flow problem," *Operations Research*, vol. 64, n°. 6, p. 1177–1196, 2016.

[37] ——, "Matrix minor reformulation and socp-based spatial branch-and-cut method for the ac optimal power flow problem," *Mathematical Programming Computation*, vol. 10, n°. 4, p. 557–596, 2018.

[38] C. Coffrin, H. L. Hijazi et P. Van Hentenryck, "The qc relaxation : A theoretical and computational study on optimal power flow," *IEEE Transactions on Power Systems*, vol. 31, n°. 4, p. 3008–3018, 2015.

[39] M. R. Narimani, D. K. Molzahn et M. L. Crow, "Tightening qc relaxations of ac optimal power flow problems via complex per unit normalization," *IEEE Transactions on Power Systems*, 2020.

[40] K. Sundar, H. Nagarajan, S. Misra, M. Lu, C. Coffrin et R. Bent, "Optimization-based bound tightening using a strengthened qc-relaxation of the optimal power flow problem," *arXiv preprint arXiv :1809.04565*, 2018.

[41] F. Zohrizadeh, C. Josz, M. Jin, R. Madani, J. Lavaei et S. Sojoudi, "A survey on conic relaxations of optimal power flow problem," *European journal of operational research*, 2020.

[42] J. B. Lasserre, "Inverse polynomial optimization," *Mathematics of Operations Research*, vol. 38, n°. 3, p. 418–436, 2013.

[43] D. K. Molzahn, B. C. Lesieutre et C. L. DeMarco, "A sufficient condition for global optimality of solutions to the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 29, n°. 2, p. 978–979, 2014.

[44] S. Xu, R. Ma, D. K. Molzahn, H. Hijazi et C. Josz, "Verifying global optimality of candidate solutions to polynomial optimization problems using a determinant relaxation hierarchy," *arXiv preprint arXiv :2101.00621*, 2021.

[45] M. Lu, H. Nagarajan, R. Bent, S. D. Eksioglu et S. J. Mason, "Tight piecewise convex relaxations for global optimization of optimal power flow," dans *2018 Power Systems Computation Conference (PSCC)*. IEEE, 2018, p. 1–7.

[46] H. Nagarajan, M. Lu, S. Wang, R. Bent et K. Sundar, "An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs," *Journal of Global Optimization*, vol. 74, n°. 4, p. 639–675, 2019.

[47] S. Gopinath, H. Hijazi, T. Weisser, H. Nagarajan, M. Yetkin, K. Sundar et R. Bent, "Proving global optimality of acopf solutions," *Electric Power Systems Research*, vol. 189, p. 106688, 2020.

[48] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on optimization*, vol. 11, n°. 3, p. 796–817, 2001.

[49] C. Josz, J. Maeght, P. Panciatici et J. C. Gilbert, "Application of the moment-sos approach to global optimization of the opf problem," *IEEE Transactions on Power Systems*, vol. 30, n°. 1, p. 463–470, 2015.

[50] X. Kuang, B. Ghaddar, J. Naoum-Sawaya et L. F. Zuluaga, "Alternative lp and socp hierarchies for acopf problems," *IEEE Transactions on Power Systems*, vol. 32, n°. 4, p. 2828–2836, 2017.

[51] D. K. Molzahn et L. A. Roald, "Towards an ac optimal power flow algorithm with robust feasibility guarantees," dans *2018 Power Systems Computation Conference (PSCC)*. IEEE, 2018, p. 1–7.

[52] ARPA-E, "Grid optimization competition," https ://gocompetition.energy.gov/, nov. 2018.

[53] A. L. L. Samuel Burer, "Non-convex mixed-integer nonlinear programming : A survey," *Surveys in Operations Research and Management Science*, vol. 17, p. 97–106, juill. 2012.

[54] T. Berthold, *Heuristic algorithms in global MINLP solvers*. Verlag Dr. Hut Munich, 2014.

[55] G. Nannicini, P. Belotti et L. Liberti, "A local branching heuristic for minlps," *arXiv preprint arXiv :0812.2188*, 2008.

[56] C. D'Ambrosio, A. Frangioni, L. Liberti et A. Lodi, "Experiments with a feasibility pump approach for nonconvex minlps," dans *International Symposium on Experimental Algorithms*. Springer, 2010, p. 350–360.

[57] P. Belotti et T. Berthold, "Three ideas for a feasibility pump for nonconvex minlp," *Optimization Letters*, vol. 11, n°. 1, p. 3–15, 2017.

[58] L. Schewe et M. Schmidt, "Computing feasible points for binary minlps with mpecs," *Mathematical Programming Computation*, vol. 11, n°. 1, p. 95–118, 2019.

[59] H. Wolkowicz, R. Saigal et L. Vandenberghe, *Handbook of semidefinite programming : theory, algorithms, and applications*. Springer Science & Business Media, 2012, vol. 27.

[60] M. F. Anjos et J. B. Lasserre, "Introduction to semidefinite, conic and polynomial optimization," dans *Handbook on semidefinite, conic and polynomial optimization*. Springer, 2012, p. 1–22.

[61] S. Bellavia, J. Gondzio et M. Porcelli, "An inexact dual logarithmic barrier method for solving sparse semidefinite programs," *Mathematical Programming*, p. 1–35, 2018.

[62] L. Vandenberghe, M. S. Andersen *et al.*, "Chordal graphs and semidefinite optimization," *Foundations and Trends in Optimization*, vol. 1, n°. 4, p. 241–433, 2015.

[63] M. Yannakakis, "Computing the minimum fill-in is np-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, n°. 1, p. 77–79, 1981.

[64] P. Heggernes, "Minimal triangulations of graphs : A survey," *Discrete Mathematics*, vol. 306, n°. 3, p. 297–317, 2006.

[65] D. Bergman, C. H. Cardonha, A. A. Cire et A. U. Raghunathan, "On the minimum chordal completion polytope," *Operations Research*, vol. 67, n°. 2, p. 532–547, 2019.

[66] P. R. Amestoy, T. A. Davis et I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, n°. 4, p. 886–905, 1996.

[67] R. E. Tarjan et M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM Journal on computing*, vol. 13, n°. 3, p. 566–579, 1984.

[68] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, n°. 6, p. 1389–1401, 1957.

[69] F. Permenter et P. Parrilo, "Partial facial reduction : simplified, equivalent sdps via approximations of the psd cone," *Mathematical Programming*, vol. 171, n°. 1, p. 1–54, 2018.

[70] D. Drusvyatskiy et H. Wolkowicz, *The Many Faces of Degeneracy in Conic Optimization*. Now Foundations and Trends, 2017.

[71] C. Helmberg et F. Rendl, "A spectral bundle method for semidefinite programming," *SIAM Journal on Optimization*, vol. 10, n°. 3, p. 673–696, 2000.

[72] R. Madani, A. Kalbat et J. Lavaei, "Admm for sparse semidefinite programming with applications to optimal power flow problem," dans *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, p. 5932–5939.

[73] ——, "A low-complexity parallelizable numerical algorithm for sparse semidefinite programming," *IEEE Transactions on Control of Network Systems*, 2017.

[74] A. Kalbat et J. Lavaei, "A fast distributed algorithm for decomposable semidefinite programs," dans *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*. IEEE, 2015, p. 1742–1749.

[75] S. Mhanna, A. C. Chapman et G. Verbič, "Component-based dual decomposition methods for the opf problem," *Sustainable Energy, Grids and Networks*, vol. 16, p. 91–110, 2018.

[76] P. Kleniati, P. Parpas et B. Rustem, "Decomposition-based method for sparse semidefinite relaxations of polynomial optimization problems," *Journal of optimization theory and applications*, vol. 145, n°. 2, p. 289–310, 2010.

[77] M. X. Goemans et D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, n°. 6, p. 1115–1145, 1995.

[78] J. B. Lasserre, “Global optimization with polynomials and the problem of moments,” *SIAM Journal on optimization*, vol. 11, n°. 3, p. 796–817, 2001.

[79] C. Josz et D. Henrion, “Strong duality in lasserre’s hierarchy for polynomial optimization,” *Optimization Letters*, vol. 10, n°. 1, p. 3–10, 2016.

[80] D. K. Molzahn et I. A. Hiskens, “Sparsity-exploiting moment-based relaxations of the optimal power flow problem,” *IEEE Transactions on Power Systems*, vol. 30, n°. 6, p. 3168–3180, 2015.

[81] T. Chen, J.-B. Lasserre, V. Magron et E. Pauwels, “A sublevel moment-sos hierarchy for polynomial optimization,” *arXiv preprint arXiv :2101.05167*, 2021.

[82] J. Wang, V. Magron et J.-B. Lasserre, “Tssos : A moment-sos hierarchy that exploits term sparsity,” *SIAM Journal on Optimization*, vol. 31, n°. 1, p. 30–58, 2021.

[83] J. Bezanson, A. Edelman, S. Karpinski et V. B. Shah, “Julia : A fresh approach to numerical computing,” *SIAM review*, vol. 59, n°. 1, p. 65–98, 2017.

[84] I. Dunning, J. Huchette et M. Lubin, “Jump : A modeling language for mathematical optimization,” *SIAM Review*, vol. 59, n°. 2, p. 295–320, 2017.

[85] R. Fourer, D. M. Gay et B. W. Kernighan, “A modeling language for mathematical programming,” *Management Science*, vol. 36, n°. 5, p. 519–554, 1990.

[86] M. Grant et S. Boyd, “CVX : Matlab software for disciplined convex programming, version 2.1,” http://cvxr.com/cvx, mars 2014.

[87] ——, “Graph implementations for nonsmooth convex programs,” dans *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd et H. Kimura, édit. Springer-Verlag Limited, 2008, p. 95–110, http://stanford.edu/~boyd/graph_dcp.html.

[88] M. Udell, K. Mohan, D. Zeng, J. Hong, S. Diamond et S. Boyd, “Convex optimization in Julia,” *SC14 Workshop on High Performance Technical Computing in Dynamic Languages*, 2014.

[89] C. Coffrin, R. Bent, K. Sundar, Y. Ng et M. Lubin, “Powermodels. jl : An open-source framework for exploring power flow formulations,” dans *2018 Power Systems Computation Conference (PSCC)*. IEEE, 2018, p. 1–8.

[90] J. C. Gilbert et C. Josz, “Plea for a semidefinite optimization solver in complex numbers - the full report,” *[Research Report] INRIA Paris ; LAAS. 2017, pp.46. hal-01497173*, 2017.

[91] R. H. Byrd, J. Nocedal et R. A. Waltz, “K nitro : An integrated package for nonlinear optimization,” dans *Large-scale nonlinear optimization*. Springer, 2006, p. 35–59.

[92] ARPA-E, “Datasets,” https ://gocompetition.energy.gov/challenges/1/datasets, nov. 2018.

[93] J. Sliwak, M. Ruiz, M. F. Anjos, L. Létocart et E. Traversi, “A julia module for polynomial optimization with complex variables applied to optimal power flow,” dans *2019 IEEE Milan PowerTech*, June 2019, p. 1–6.

[94] A. Eltved, J. Dahl et M. S. Andersen, “On the robustness and scalability of semidefinite relaxation for optimal power flow problems,” *Optimization and Engineering*, vol. 21, n°. 2, p. 375–392, 2020.

[95] R. A. Jabr, “Exploiting sparsity in sdp relaxations of the opf problem,” *IEEE Transactions on Power Systems*, vol. 27, n°. 2, p. 1138–1139, 2012.

[96] D. J. Rose, “Triangulated graphs and the elimination process,” *Journal of Mathematical Analysis and Applications*, vol. 32, n°. 3, p. 597–609, 1970.

[97] ——, “A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations,” dans *Graph theory and computing*. Elsevier, 1972, p. 183–217.

[98] S. Bromberger, J. Fairbanks et other contributors, “Juliagraphs/lightgraphs.jl : an optimized graphs package for the julia programming language,” 2017. [En ligne]. Disponible : https://doi.org/10.5281/zenodo.889971

[99] B. Borchers, “Csdp, ac library for semidefinite programming,” *Optimization methods and Software*, vol. 11, n°. 1-4, p. 613–623, 1999.

[100] Mosek, “The mosek optimization software,” *Online at http ://www. mosek. com*, 2019.

[101] B. O’Donoghue, E. Chu, N. Parikh et S. Boyd, “Operator splitting for conic optimization via homogeneous self-dual embedding,” *arXiv preprint arXiv :1312.3039*, vol. 1, n°. 2.1, p. 3, 2013.

[102] J. F. Sturm, “Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones,” *Optimization methods and software*, vol. 11, n°. 1-4, p. 625–653, 1999.

[103] A. Berry, P. Heggernes et G. Simonet, “The minimum degree heuristic and the minimal triangulation process,” dans *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 2003, p. 58–70.

[104] K. Fujisawa, M. Fukuda et K. Nakata, “Preprocessing sparse semidefinite programs via matrix completion,” *Optimization Methods and Software*, vol. 21, n°. 1, p. 17–39, 2006.

[105] Y. Sun, M. S. Andersen et L. Vandenberghe, “Decomposition in conic optimization with partially separable structure,” *SIAM Journal on Optimization*, vol. 24, n°. 2, p. 873–897, 2014.

[106] J. Sliwak, M. Anjos, L. Létocart, J. Maeght et E. Traversi, “Improving clique decompositions of semidefinite relaxations for optimal power flow problems,” EasyChair Preprint no. 2546, EasyChair, 2020.

[107] J. Sliwak, E. D. Andersen, M. F. Anjos, L. Létocart et E. Traversi, “A clique merging algorithm to solve semidefinite relaxations of optimal power flow problems,” *IEEE Transactions on Power Systems*, vol. 36, n°. 2, p. 1641–1644, 2021.

[108] H. Waki, S. Kim, M. Kojima et M. Muramatsu, “Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity,” *SIAM Journal on Optimization*, vol. 17, n°. 1, p. 218–242, 2006.

[109] B. Zhou, K. W. Chan, T. Yu, H. Wei et J. Tang, “Strength pareto multigroup search optimizer for multiobjective optimal reactive power dispatch,” *IEEE Transactions on Industrial Informatics*, vol. 10, n°. 2, p. 1012–1022, 2014.

[110] K. Lenin, B. R. Reddy et M. Suryakalavathi, “Hybrid tabu search-simulated annealing method to solve optimal reactive power problem,” *International Journal of Electrical Power & Energy Systems*, vol. 82, p. 87–91, 2016.

[111] A. Rajan et T. Malakar, “Exchange market algorithm based optimum reactive power dispatch,” *Applied Soft Computing*, vol. 43, p. 320–336, 2016.

[112] M. Mehdinejad, B. Mohammadi-Ivatloo, R. Dadashzadeh-Bonab et K. Zare, “Solution of optimal reactive power dispatch of power systems using hybrid particle swarm optimization and imperialist competitive algorithms,” *International Journal of Electrical Power & Energy Systems*, vol. 83, p. 104–116, 2016.

[113] A. A. Heidari, R. A. Abbaspour et A. R. Jordehi, “Gaussian bare-bones water cycle algorithm for optimal reactive power dispatch in electrical power systems,” *Applied Soft Computing*, vol. 57, p. 657–671, 2017.

[114] K. ben oualid Medani, S. Sayah et A. Bekrar, “Whale optimization algorithm based optimal reactive power dispatch : A case study of the algerian power system,” *Electric Power Systems Research*, vol. 163, p. 696–705, 2018.

[115] U. Kılıç, K. Ayan et U. Arifoğlu, “Optimizing reactive power flow of hvdc systems using genetic algorithm,” *International Journal of Electrical Power & Energy Systems*, vol. 55, p. 1–12, 2014.

[116] G. Chen, L. Liu, Z. Zhang et S. Huang, “Optimal reactive power dispatch by improved gsa-based algorithm with the novel strategies to handle constraints,” *Applied Soft Computing*, vol. 50, p. 58–70, 2017.

[117] H. Xu, A. Dominguez-Garcia et P. W. Sauer, "Optimal tap setting of voltage regulation transformers using batch reinforcement learning," *IEEE Transactions on Power Systems*, 2019.

[118] C. Bingane, M. F. Anjos et S. Le Digabel, "Tight-and-cheap conic relaxation for the optimal reactive power dispatch problem," *IEEE Transactions on Power Systems*, vol. 34, nº. 6, p. 4684–4693, 2019.

[119] S. E. Kayacik et B. Kocuk, "An misocp-based solution approach to the reactive optimal power flow problem," *IEEE Transactions on Power Systems*, 2020.

[120] H. Hijazi, C. Coffrin et P. Van Hentenryck, "Convex quadratic relaxations for mixed-integer nonlinear programs in power systems," *Mathematical Programming Computation*, vol. 9, nº. 3, p. 321–367, 2017.

[121] F. Capitanescu et L. Wehenkel, "Sensitivity-based approaches for handling discrete variables in optimal power flow computations," *IEEE Transactions on Power Systems*, vol. 25, nº. 4, p. 1780–1789, 2010.

[122] Z. Yang, A. Bose, H. Zhong, N. Zhang, Q. Xia et C. Kang, "Optimal reactive power dispatch with accurately modeled discrete control devices : A successive linear approximation approach," *IEEE Transactions on Power Systems*, vol. 32, nº. 3, p. 2435–2444, 2017.

[123] E. Davoodi, E. Babaei, B. Mohammadi-Ivatloo et M. Rasouli, "A novel fast semidefinite programming-based approach for optimal reactive power dispatch," *IEEE Transactions on Industrial Informatics*, vol. 16, nº. 1, p. 288–298, 2019.

[124] J. C. Lopez, M. J. Rider *et al.*, "Optimal reactive power dispatch with discrete controllers using a branch-and-bound algorithm : A semidefinite relaxation approach," *IEEE Transactions on Power Systems*, 2021.

[125] M. Fischetti et A. Lodi, "Local branching," *Mathematical programming*, vol. 98, nº. 1, p. 23–47, 2003.

[126] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs : Part i—convex underestimating problems," *Mathematical programming*, vol. 10, nº. 1, p. 147–175, 1976.

[127] N. V. Sahinidis, "Global optimization and constraint satisfaction : The branch-and-reduce approach," dans *International Workshop on Global Optimization and Constraint Satisfaction*. Springer, 2002, p. 1–16.

[128] J. Sliwak, M. F. Anjos, L. Létocart et E. Traversi, "A Semidefinite Optimization-based Branch-and-Bound Algorithm for Several Reactive Optimal Power Flow

Problems," mars 2021, working paper or preprint. [En ligne]. Disponible : https://hal.archives-ouvertes.fr/hal-03179726

[129] R. Louca et E. Bitar, "Robust ac optimal power flow," *IEEE Transactions on Power Systems*, vol. 34, nº. 3, p. 1669–1681, 2018.

[130] D. Phan et S. Ghosh, "Two-stage stochastic optimization for optimal power flow under renewable generation uncertainty," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 24, nº. 1, p. 1–22, 2014.

[131] L. Roald et G. Andersson, "Chance-constrained ac optimal power flow : Reformulations and efficient algorithms," *IEEE Transactions on Power Systems*, vol. 33, nº. 3, p. 2906–2918, 2017.

[132] M. Lubin, Y. Dvorkin et L. Roald, "Chance constraints for improving the security of ac optimal power flow," *IEEE Transactions on Power Systems*, vol. 34, nº. 3, p. 1908–1917, 2019.