INSTITUT POLYTECHNIQUE DE PARIS

NNT : 20XXIPPAXXXX

Thèse de doctorat

Relaxation methods for mixed-integer nonlinear programming

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Palaiseau, le Date, par

LIDING XU

Composition du Jury :

François Clautiaux Professeur, Institut de Mathématiques de Bordeaux, Université de Bordeaux & Inria Bordeaux.	Président
Ruth Misener Professeur, Department of Computing, Imperial College London	Rapporteur
Michaël Poss Directeur de Recherche, LIRMM CNRS, Université de Montpellier	Rapporteur
Roberto Wolfler-Calvo Professeur, LIPN CNRS, Université de Paris-Nord	Examinateur
Leo Liberti Directeur de Recherche, LIX CNRS, École polytechnique, Institut Polytechnique de Paris	Directeur de thèse
Sonia Haddad-Vanier Maître de Conférence HDR, Université Panthéon-Sorbonne & LIX CNRS, École polytechnique, Institut Polytechnique de Paris	Co-directeur de thèse
Claudia D'Ambrosio Directrice de Recherche, LIX CNRS, École polytechnique, Institut Polytechnique de Paris	Invité
Emiliano Traversi Professeur, LIRMM CNRS, University of Montpellier	Invité

I would like to dedicate this thesis to my loving parents . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This thesis contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Liding Xu
December 2023

Acknowledgements

I am profoundly indebted to my supervisors, Leo Liberti, Claudia D'Ambrosio, and Sonia Haddad-Vanier for their unwavering support, guidance, and mentorship throughout my doctoral journey. Their expertise and encouragement have been invaluable. Additionally, this endeavor would not have been possible without the generous support from École doctorale IP Paris and the French taxpayers, who financed my research.

Words cannot express my gratitude to my defense committee. I could not have undertaken this journey without my defense committee, who generously provided knowledge and expertise.

I am also grateful to my colleagues for their discussion and inspiration. Thanks should also go to the administrative staff who helped me a lot with administration issues.

Lastly, I should mention my family, especially my parents. Their belief in me has kept my spirits and motivation high during this process. I would also like to thank my girlfriend for all the emotional support.

Abstract

This thesis focuses on mixed-integer nonlinear programming (MINLP), a class of mathematical optimization problems, and the associated algorithms to solve them. The core algorithm utilized in many global optimization solvers for MINLP problems is the branch-and-bound algorithm. Key to the success of the branch-and-bound approach is the use of relaxations of optimization problems, which are vital in obtaining efficient and tight dual bounds. However, constructing effective relaxations depends on the specific structures of optimization problems. In the first part of this thesis, we present a comprehensive overview of structural relaxation tools tailored for structured MINLP problems across different disciplines. These tools encompass relaxations from extended formulations, relaxations via submodularity, relaxations using piece-wise linear approximation, and relaxation tightening via intersection cuts. Then, we develop novel advanced theoretical results based on these tools. In the second part, we employ these relaxation techniques to address various optimization problems. We explore cutting planes for signomial programming. Then, we propose intersection cuts for enhancing linear programming relaxations of submodular optimization problems. Next, we investigate the Dantzig-Wolfe relaxations for a mixed-integer linear programming problem in wireless network routing and a MINLP problem in submodular binpacking. Finally, we study the big-M relaxation technique as applied to piece-wise linear functions in the continuous covering problem on a network. By combining these comprehensive studies on various relaxation techniques and their applications in different optimization contexts, this thesis contributes to the advancement of MINLP and related optimization methods, offering valuable insights for both theoretical understanding and computational implementation.

Résumé

Cette thèse se concentre sur la programmation non linéaire à variables mixtes (MINLP), une classe de problèmes d'optimisation mathématique, et les algorithmes associés pour les résoudre. L'algorithme central utilisé dans de nombreux solveurs d'optimisation globale pour les problèmes MINLP est l'algorithme de séparation et évaluation. La clé du succès de l'algorithme de séparation et évaluation réside dans l'utilisation de relaxations des problèmes d'optimisation, qui sont essentielles pour obtenir des bornes duales efficaces. Cependant, la construction de relaxations efficaces dépend des structures spécifiques des problèmes d'optimisation. Dans la première partie de cette thèse, nous présentons un aperçu complet des outils de relaxation structurelle adaptés aux problèmes MINLP structurés liés à différents domaines d'applications. Ces outils englobent des relaxations à partir de formulations étendues, des relaxations par sous-modularité, des relaxations utilisant une approximation linéaire par morceaux et des renforcements de relaxation via des coupes d'intersection. Nous développons de nouveaux résultats théoriques avancés basés sur ces outils. Dans la deuxième partie, nous utilisons ces techniques de relaxation pour aborder divers problèmes d'optimisation. Nous explorons les plans coupants pour la programmation signoïdale. Nous proposons des coupes d'intersection pour améliorer les relaxations linéaire des problèmes d'optimisation sous-modulaire. Nous étudions les relaxations de Dantzig-Wolfe pour un problème de programmation linéaire à variables mixtes dans le routage de réseaux sans fil et un problème MINLP dans le binpacking sous-modulaire. Enfin, nous étudions la technique de relaxation big-M appliquée aux fonctions linéaires par morceaux dans le problème de couverture continue sur un réseau. Les travaux réalisés durant cette thèse de doctorat contribuent à l'avancement des approches de la programmation non linéaire en nombre entiers et des méthodes d'optimisation connexes. En effet la combinaison des études exhaustives réalisées sur diverses techniques de relaxation et leurs applications à différents contextes d'optimisation offrent des perspectives précieuses tant pour la compréhension théorique des problèmes que pour la mise en œuvre empirique des résultats.

Table of contents

List of figures **xvii**

List of tables **xix**

1 Introduction en français **1**
1.1 Classifications des problèmes MINLP . . . 3
1.2 Optimisation MINLP basée sur la relaxation . . . 6
1.3 Structure de la thèse . . . 8
1.4 Publications antérieures . . . 9

2 Introduction **11**
2.1 Classifications of MINLP problems . . . 13
2.2 Relaxation-based MINLP optimization . . . 15
2.3 Structure of the thesis . . . 18
2.4 Prior publications . . . 18

3 Theory: basic relaxation methods **19**
3.1 Relaxations via lifting . . . 19
3.1.1 Dantzig-Wolfe relaxation . . . 20
3.1.2 Projection . . . 22
3.1.3 Factorable programming . . . 23
3.1.4 Certificate-based relaxation . . . 24
3.2 Relaxations via submodularity . . . 26
3.2.1 Convex envelope . . . 27
3.2.2 Concave overestimator . . . 28
3.3 Relaxations via piece-wise linear approximations . . . 29
3.4 Relaxation tightening via intersection cuts . . . 31
3.5 Conclusion . . . 33

4 Theory: advanced structural results **35**
4.1 $\mathcal{S}$-free sets for structured sets . . . 35
4.1.1 Maximal $\mathcal{S}$-free sets for lifted sets . . . 35
4.1.2 Maximal $\mathcal{S}$-free for DCC constraints . . . 37
4.1.3 $\mathcal{S}$-free sets in submodular optimization . . . 40
4.2 Convex envelopes of supermodular functions . . . 50

5 Cutting planes for signomial programming **53**
5.1 Introduction . . . 53
5.1.1 Literature review . . . 54
5.1.2 Contribution . . . 55
5.1.3 Notation . . . 56
5.1.4 Outline of the chapter . . . 56
5.2 Signomial-lift-free sets and intersection cuts . . . 56
5.2.1 Signomial-lift-free and signomial-term-free sets . . . 56
5.2.2 Computing intersection cuts . . . 59
5.3 Convex outer approximation . . . 60
5.3.1 Outer approximation cuts . . . 61
5.3.2 Convex envelopes of bivariate supermodular functions . . . 62
5.3.3 Convexity and reverse-convexity . . . 63
5.4 Computational results . . . 63
5.5 Conclusion . . . 65

6 Intersection cuts for submodular optimization **67**
6.1 Introduction . . . 67
6.1.1 Literature review . . . 67
6.1.2 Contribution . . . 68
6.1.3 Outline of the chapter . . . 69
6.2 Application . . . 69
6.2.1 Boolean multilinear constraints . . . 69
6.2.2 D-optimal design . . . 72
6.3 Separation problem . . . 73
6.4 Computational results . . . 76
6.4.1 Supplementary branch-and-bound computational results . . . 83
6.5 Conclusion . . . 84

7 Branch-and-price for submodular bin packing **85**
7.1 Introduction . . . 85
7.1.1 Literature review . . . 88
7.1.2 Contribution . . . 90
7.1.3 Outline of the chapter . . . 91
7.2 Set cover formulation . . . 91
7.3 Branch and price . . . 92
7.3.1 Branching rule . . . 93
7.3.2 Column generation . . . 93
7.3.3 Primal heuristics . . . 95
7.3.4 Dual bound computation . . . 95
7.4 Solving the pricing problem . . . 95
7.4.1 Pricing heuristic . . . 96
7.4.2 BSOCP formulation . . . 96
7.4.3 MBQCP formulation . . . 99
7.4.4 PWL relaxation . . . 100
7.4.5 Exact PWL-B&C algorithm . . . 103

7.4.6 Hybrid pricing strategy . . . 106
7.5 Computational experiments . . . 107
7.5.1 Benchmarks . . . 107
7.5.2 Experimental setups . . . 109
7.5.3 Comparative analysis of results . . . 111
7.6 Conclusion . . . 113

8 Branch-and-price for coding-aware routing in wireless networks 115
8.1 Introduction . . . 115
8.1.1 Literature review . . . 116
8.1.2 Contribution . . . 116
8.1.3 Outline the chapter . . . 117
8.2 Models and notation . . . 117
8.2.1 Energy consumption . . . 119
8.2.2 Clique capacity constraint . . . 119
8.2.3 Network coding . . . 121
8.2.4 Complexity analysis . . . 124
8.3 Mathematical Formulations . . . 124
8.3.1 Compact edge-based formulations . . . 124
8.3.2 Path-based formulation . . . 128
8.4 Column generation . . . 129
8.4.1 Reduced cost pricing . . . 129
8.4.2 Farkas pricing . . . 136
8.5 Branching rule . . . 137
8.6 Computational Results . . . 139
8.6.1 Instances . . . 139
8.6.2 Numerical comparisons of formulations . . . 141
8.6.3 Effects of network coding . . . 142
8.7 Conclusion . . . 144

9 Piece-wise linear modelling in continuous covering on networks 147
9.1 Introduction . . . 147
9.1.1 Literature review . . . 150
9.1.2 Contribution . . . 151
9.1.3 Outline the chapter . . . 151
9.2 Covering characterization . . . 152
9.2.1 Observations . . . 152
9.2.2 Covering conditions . . . 153
9.2.3 Covering delimitation and simplification . . . 154
9.2.4 Characterization of complete covers . . . 157
9.3 MILP formulation . . . 159
9.3.1 Comparative insights with respect to an existing MILP . . . 161
9.4 Strengthening . . . 162
9.4.1 Constants tightening . . . 162
9.4.2 Valid inequalities . . . 163
9.5 Network processing . . . 165

9.6 A reduced formulation for networks with long edges . . . 170
9.7 Computational results . . . 173
9.7.1 Experiment Setup . . . 173
9.7.2 Comparative Analysis of Continuous Models . . . 176
9.7.3 Comparative Analysis of Continuous and Discrete Models . . . 177
9.8 Conclusions . . . 180

10 Conclusions and perspective **183**
10.0.1 Conclusions . . . 183
10.1 Perspectives in algorithm design . . . 184
10.1.1 Extended formulations vs projected formulations . . . 184
10.1.2 Modeling power: the lack of convexity and nonlinear function types . . . 185
10.1.3 Approximations vs. relaxations . . . 186
10.2 Perspectives of solver framework . . . 186

References **189**

List of figures

3.1 PWL approximation of nonconvex sets. 31
3.2 An $\mathcal{S}$-free set $\mathcal{C}$. 32
3.3 An $\mathcal{S}$-free set $\mathcal{C}$, simplicial cone $\mathcal{R}$, and intersection cut. 33

4.1 a Boolean cube $\mathcal{B} = \{0,1\}^3$ © [134] 47

5.1 $\mathcal{S}_{\mathrm{st}}$-free sets. 59
5.2 Two examples of $\mathcal{S}_{\mathrm{st}}$ and $\mathcal{S}_{\mathrm{st}}$-free sets. 59

6.1 Scatter plots of MAX CUT results in standalone (top) and embedded (bottom) configurations 79
6.2 Scatter plots of PSEUDO BOOLEAN MAXIMIZATION results in standalone (top) and embedded (bottom) configurations 80
6.3 Scatter plots of BAYESIAN D-OPTIMAL DESIGN results in standalone (top) and embedded (bottom) configurations 82
6.4 Scatter of MAX CUT-sub results in the embedded branch-and-bound test 84

7.1 Graphs of the quadratic and its PWL over-estimator 100

8.1 Interference model 120
8.2 Network coding 121
8.3 G and EG 131
8.4 π^{-1} does not preserve the acyclicity from EG to G 132
8.5 Paths diverge at i and k 138

9.1 Two cycle coverage points with respect to a cycle C of five nodes 148
9.2 An instance of CSCP such that not all facilities in $\mathcal{P}^*$ are at a distance δ from some node 150
9.3 Covering of an edge $e = (v_a, v_b) \in E$ 154
9.4 Illustration of valid inequalities (9.9) 164
9.5 Illustration of valid inequalities (9.10) 165
9.6 Illustration of Prop. 9.21 for a long edge e 170
9.7 Scatter plots of the relative dual gaps and the relative primal bounds between different settings 179
9.8 Scatter plots of the relative dual gaps and the relative primal bounds between **SFD** and **RF** 180

List of tables

5.1 Summary of performance metrics on MINLPLib instances 64

6.1 Summary of MAX CUT results 79
6.2 Summary of PSEUDO BOOLEAN MAXIMIZATION results 80
6.3 Summary of BAYESIAN D-OPTIMAL DESIGN results 82
6.4 Summary of MAX CUT-sub results in the embedded branch-and-bound test . . 83

7.1 Example distribution of item size 108
7.2 Aggregated statistics of the main computational results 111
7.3 Master and pricing problem statistics of different configurations 113

8.1 Comparison of variables 129
8.2 Comparison of constraints 129
8.3 Solver performance on low-demand test bed 142
8.4 Solver performance on high-demand test bed 143
8.5 Effects of network coding on low-demand test bed 144
8.6 Effects of network coding on high-demand test bed 145

9.1 Comparative summary on MILP formulations for the CSCP 162
9.2 Model summary 175
9.3 Results for continuous models 178
9.4 Results for continuous and discrete models 179

Chapter 1

Introduction en français

La pratique conventionnelle des mathématiques appliquées repose sur trois processus essentiels : la modélisation, la simulation et l'optimisation.

Les scientifiques sont généralement d'avis qu'il existe des principes bien définis régissant les processus naturels et humains. La modélisation est le domaine consacré à la découverte et à la description de ces principes au moyen de langages formels. Les langages formels peuvent être analysés par les ordinateurs en code exécutable. Dans cette thèse, nous considérons des modèles mathématiques décrits formellement par des entrées connues, des entrées inconnues et éventuellement des critères de décision pour les sorties.

La majorité des processus naturels et humains, tels que les phénomènes physiques, ne sont pas entièrement sous le contrôle de l'homme ou sont excessivement coûteux à reproduire. La simulation crée des environnements permettant d'imiter ces processus à l'aide de leurs modèles mathématiques, facilitant ainsi la caractérisation de leurs propriétés. Les ordinateurs contemporains permettent de créer un environnement virtuel en traduisant les modèles du langage formel en codes exécutables, puis en les exécutant pour produire des données de sortie. Aujourd'hui, grâce à de puissantes ressources informatiques, la simulation peut désormais traiter des modèles mathématiques avec un grand nombre d'entrées, comme la simulation de systèmes quantiques et de systèmes multi-agents.

L'optimisation est le processus mathématique qui consiste à trouver les valeurs des sorties inconnues d'un modèle qui satisfont à des critères donnés au moyen d'entrées observées. Dans ce contexte, le modèle mathématique est appelé "modèle d'optimisation" et le problème "problème d'optimisation". La procédure d'optimisation est généralement appelée *algorithme*, qui peut être mis en œuvre sur ordinateur. L'optimisation joue un rôle crucial dans l'ajustement des modèles et la prise de décision, comme dans le problème du voyageur de commerce, où les bons résultats correspondent à des itinéraires peu coûteux.

Bien que la simulation et l'optimisation utilisent toutes deux des ordinateurs pour exécuter des modèles mathématiques, leurs objectifs peuvent diverger. La simulation vise à reproduire les processus naturels et humains sur la base de leurs modèles mathématiques, tandis que l'optimisation part du principe que les modèles mathématiques fournis sont toujours valables et se concentre sur la recherche de solutions optimales.

La programmation mathématique (PM) est un langage formel spécifique permettant de décrire la plupart des problèmes d'optimisation. Chaque phrase formelle est appelée une *formulation* d'un problème d'optimisation donné. Chaque formulation MP décompose un

problème d'optimisation (ou une classe de problèmes) en cinq entités symboliques, appelées : (i) paramètres, qui codent l'entrée du problème ; (ii) variables de décision, qui codent la sortie du problème ; (iii) une ou plusieurs fonctions objectives, exprimées en termes de paramètres et de variables de décision, qui codent les critères à optimiser ; (iv) zéro ou plusieurs contraintes explicites, qui sont des critères dépendant des paramètres et des variables de décision ; (v) zéro ou plusieurs contraintes implicites, qui sont des critères énonçant l'appartenance de certaines variables de décision à un ensemble donné. Les fonctions et les contraintes explicites sont explicitement données en termes d'expressions mathématiques, tandis que l'ensemble apparaissant dans les contraintes implicites doit être pris en compte par un algorithme de solution déployé sur la formulation.

L'optimisation dans l'incertitude est l'étude des problèmes d'optimisation lorsqu'il y a une incertitude concernant les paramètres impliqués. Dans le cas général, le traitement de cette incertitude implique l'incorporation de méthodes d'échantillonnage dans un processus d'optimisation, permettant une approximation probabiliste des problèmes d'optimisation par le biais de l'échantillonnage. Toutefois, il convient de noter que cette thèse n'abordera pas ces questions.

Les algorithmes de solution pour les formulations MP sont appelés *solvers*. Ces solveurs correspondent à une taxonomie de classes de formulations MP organisées autour de la continuité ou de l'intégralité des variables de décision, ainsi que de la linéarité, de la non-linéarité, de la convexité de la (des) fonction(s) objective(s) et des contraintes explicites. Par exemple, les formulations MP avec des variables entières et des formes linéaires sont appelées Programmation linéaire en nombres entiers mixtes (MILP). Lorsque les formulations MP comprennent à la fois des variables entières et des termes non linéaires, elles entrent dans la catégorie connue sous le nom de programmation non linéaire à nombre entier mixte (MINLP).

L'optimisation globale est l'étude des problèmes d'optimisation impliquant la non-linéarité et la non-convexité. Il s'agit d'une classe plus large que la MINLP, car elle comprend également les problèmes d'optimisation dits "boîte noire", où les fonctions sont données comme des oracles, plutôt que comme des expressions mathématiques - mais cette thèse ne traitera pas de ces problèmes.

Comme nous le démontrerons plus tard, MINLP peut exprimer de nombreux types de problèmes d'optimisation. Actuellement, les solveurs MINLP [48, 62, 283] peuvent théoriquement traiter globalement plusieurs types de problèmes MINLP structurés, y compris les problèmes contraints décrits par des fonctions élémentaires (telles que les fonctions puissance et trigonométriques) et les cônes convexes élémentaires (y compris les cônes polyédriques et les cônes de second ordre). Cependant, il est crucial de noter qu'il existe des sous-classes de problèmes MINLP qui peuvent être prouvés comme étant $\mathcal{NP}$-complets ou même indécidables. Les thèmes de la complexité et de la calculabilité de MINLP, tels que discutés dans [203], sortent du cadre de cette thèse.

MINLP s'inspire de certaines méthodologies de MILP, ce qui lui permet de traiter des problèmes avec des variables de décision discrètes issus de la recherche opérationnelle et de l'optimisation combinatoire. MINLP va plus loin et étend ses capacités pour traiter également des modèles mathématiques non linéaires. Par conséquent, MINLP trouve sa pertinence dans divers domaines tels que la recherche chimique [253], l'ingénierie des processus [192], et la théorie du contrôle [172]. Cependant, l'état actuel des connaissances limite la vitesse des solveurs, ce qui laisse une place importante à la recherche.

Cette thèse étudie différentes approches pour résoudre plusieurs classes de problèmes MINLP. L'objectif principal de cette thèse est de réduire la disparité entre la gamme limitée d'algorithmes d'optimisation disponibles et les vastes problèmes MINLP. Le thème central de notre étude tourne autour des méthodes de relaxation pour les MINLP. Dans les sections suivantes, nous définirons et catégoriserons les problèmes MINLP, tout en fournissant une vue d'ensemble des techniques fondamentales d'optimisation basées sur la relaxation.

1.1 Classifications des problèmes MINLP

Un problème MINLP admet formellement la forme suivante:

$$\begin{aligned} & \inf && f_0(x) && \text{(1.1a)} \\ \text{s.t.} \quad & i \in [m] && f_i(x) \in \mathcal{S}_i && \text{(1.1b)} \\ & j \in [n] && x_j \in [\ell_j, u_j] && \text{(1.1c)} \\ & j \in [k] && x_j \in \mathbb{Z}, && \text{(1.1d)} \end{aligned}$$

où $[n] := \{1, \cdots, n\}$ est l'ensemble d'index de toutes les variables, $[k] := \{1, \cdots, k\}$ est l'ensemble d'index des variables entières, $[m] := \{1, \cdots, m\}$ est l'ensemble d'index des contraintes de la fonction dans l'ensemble. La fonction objective f_0 associe les variables à une valeur scalaire. Pour tout $i \in [m]$, $\mathcal{S}_i$ est un ensemble intégré dans un espace linéaire, et f_i associe les variables à un vecteur dans cet espace linéaire. Pour tout $j \in [n]$, les constantes scalaires ℓ_j, u_j sont respectivement les bornes inférieure et supérieure de la variable x_j, et $-\infty \leq \ell_j < u_j \leq +\infty$. Les paramètres ci-dessus sont également appelés *données* du problème MINLP.

Chaque contrainte $x_j \in \mathbb{Z}$ dans (1.1d) est appelée une *contrainte d'intégrité*, chaque contrainte $x_j \in [\ell_j, u_j]$ dans (1.1b) est appelée *contrainte de limite de variable*, et chaque contrainte $f_i(x) \in \mathcal{S}_i$ est appelée *contrainte de fonction dans l'ensemble*. Une contrainte fonction-en-ensemble est composée d'une fonction et d'un ensemble, et permet de modéliser des contraintes complexes. Par exemple, $\mathcal{S}_i$ peut être un ensemble non convexe ou un collecteur. Même s'il est possible de représenter les contraintes d'intégralité et de limite de variable comme des contraintes uniques de fonction dans un ensemble, la convention est de les exprimer individuellement.

L'étape initiale de la résolution des problèmes MINLP consiste à identifier leurs types, une procédure cruciale appliquée dans les solveurs MINLP généraux. La traçabilité des problèmes MINLP dépend de leurs types spécifiques, que nous définissons sur la base des types de variables et de contraintes qu'ils contiennent.

Les types MINLP sont des compositions de types élémentaires. Pour indiquer que le type T est un sous-type du type T', nous utilisons la notation T $\preceq$ T'. En particulier, dans de nombreux langages de programmation, nous utilisons $\varnothing$ pour désigner le type NULL afin de compléter la syntaxe. Le type NULL est le sous-type de n'importe quel type. Nous allons maintenant définir les types élémentaires et leurs sous-types.

Le type élémentaire TI a des sous-types dans {entier, binaire, $\varnothing$}, et il concerne l'aspect discret d'un problème MINLP. S'il y a au moins une variable entière ($k \neq 0$), alors TI est du sous-type "entier". Si, en outre, les bornes de toutes les variables entières se situent dans l'intervalle $[0, 1]$, alors TI est du sous-type "binaire". En l'absence de variables entières ($k = 0$),

TI est représenté par $\varnothing$ (type NULL). Il est important de noter que le type "binaire" est un sous-type du type "entier" (binaire $\preceq$ entier).

Le type élémentaire TC a des sous-types dans {linéaire, conique, non-linéaire}, et il est lié à l'aspect continu d'un problème MINLP. Si chaque fonction f_i est affine et que son ensemble associé S_i est un orthant non négatif/non positif ou un cône zéro, alors TC est du sous-type "linéaire". Si chaque fonction f_i est affine et que son ensemble associé S_i est un cône convexe, alors TC est du sous-type "conique". Cependant, si au moins une fonction f_i est non linéaire ou si au moins un ensemble $\mathcal{S}_i$ est non polyédrique, alors TC est du sous-type "non linéaire". Il est essentiel de reconnaître que le type "linéaire" est un sous-type du type "conique" (linéaire $\preceq$ conique), et que le type "conique" est un sous-type du type "non linéaire" (conique $\preceq$ non linéaire).

Le type conique possède des sous-types qui correspondent à des cônes convexes, tels que le cône polyédrique, le cône de puissance, le cône de second ordre, le cône des matrices semi-définies et le cône des matrices composites. Le type non linéaire possède également des sous-types qui ne font pas partie du type conique, tels que le type polynomial et le type signomial.

Le type élémentaire TM a des sous-types dans {mixte, $\varnothing$}, et il capture l'interaction des propriétés discrètes et continues. Lorsque certaines variables entières, mais pas toutes ($0 \neq k \neq n$), sont présentes dans le problème MINLP, le TM est du sous-type "mixte". Cependant, si toutes les variables ($k = n$) ou aucune ($k = 0$) sont entières, alors TM est du sous-type $\varnothing$ (NULL). Il convient de noter que $\varnothing \preceq$ mixte, ce qui signifie qu'un problème continu est un sous-type de problème mixte.

Definition 1.1. *Un type de MINLP est un produit type de TM,TI,TC.*

Dans un problème MINLP de types élémentaires TM, TI, TC, sa relaxation continue devient un problème de programmation non linéaire (NLP) d'un type de TC. Ainsi, chaque type MINLP a un type dérivé correspondant pour sa relaxation continue.

Definition 1.2. *Un type dérivé TD d'un type MINLP TM, TI, TC appartient à {convexe, non-convexe}. Si une instance de problème de type TC est convexe, alors le type dérivé TD de TC est convexe ; sinon, TD est non convexe.*

Généralement, les problèmes MINLP sont désignés par le terme de "programmes MTC TI TC", désignant leurs types élémentaires et leurs propriétés discrètes et continues. Parfois, nous examinons également le type dérivé TD, auquel cas nous l'appelons "programme TM TI TC". Pour simplifier les choses, nous utilisons fréquemment des abréviations basées sur une règle simple : nous conservons et mettons en majuscules les lettres initiales de TM, TI, TC et "programme", tandis que TD n'est pas abrégé.

Prenons les exemples suivants pour illustrer notre propos : Si un problème MINLP est du type "mixte, entier, linéaire", il est appelé "programme linéaire mixte en nombres entiers" et est abrégé en MILP. Si le problème MINLP est un MILP et que, de plus, son TI est binaire, il devient un "programme linéaire binaire mixte" et est abrégé en MBLP. Si le type d'un problème MINLP est "entier, polynomial", son type dérivé est non convexe. Dans ce cas, on parle de "programme polynomial en nombres entiers" (IPP) ou de "programme polynomial en nombres entiers non convexe" (IPP non convexe).

Dans la représentation alternative suivante, le problème MINLP peut être considéré comme un problème d'optimisation linéaire sur son ensemble réalisable. Pour transformer la

formulation (1.1) en cette représentation, les processus suivants peuvent être suivis : ajouter la contrainte $f(x) \leq t$, où t est une variable supplémentaire ; changer l'objectif du problème d'optimisation en $\inf t$; considérer t comme faisant partie des variables du problème. Cela donne lieu à la formulation alternative des problèmes MINLP:

$$\begin{aligned} & \inf \quad cx && \text{(1.2a)} \\ \text{s.t.} \quad & i \in [m] \quad f_i(x) \in \mathcal{S}_i && \text{(1.2b)} \\ & j \in [n] \quad x_j \in [\ell_j, u_j] && \text{(1.2c)} \\ & j \in [k] \quad x_j \in \mathbb{Z}. && \text{(1.2d)} \end{aligned}$$

Sans perte de généralité, nous nous concentrons sur les problèmes MINLP représentés dans la formulation de (1.2). Un problème MINLP est considéré comme *traçable* s'il existe un algorithme capable de l'approximer avec une précision arbitraire en un temps qui est polynomial par rapport à sa taille d'encodage et au niveau de précision souhaité. Un grand nombre de problèmes MINLP traçables sont des problèmes d'optimisation convexe (continue), et la traçabilité peut être obtenue par l'algorithme de l'ellipsoïde [132]. Cet algorithme repose sur le concept suivant d'oracle de séparation.

Definition 1.3. *Étant donné un ensemble convexe compact $K \subseteq \mathbb{R}^n$, un oracle de séparation pour K est un oracle (boîte noire) qui, étant donné un vecteur $x\ dans \mathbb{R}^n$, renvoie l'un des éléments suivants:*

- *que $x \in K$;*
- *Trouver un hyperplan qui sépare x de K : un vecteur $a \in \mathbb{R}^n$, tel que $ay > ax$ pour tout $y \in K$.*

Puisque nous considérons le problème MINLP comme dans (1.2), nous appelons la séparation de MINLP la séparation de MINLP comme la séparation de son ensemble réalisable. La classification des problèmes MINLP permet d'identifier la première sous-classe de problèmes MINLP traçables.

Lemma 1.4 ([293])**.** *Si un problème NLP convexe avec un ensemble réalisable compact possède un oracle de séparation en temps polynomial, alors il est traitable.*

La programmation linéaire (PL) et la programmation semi-définie (PDS) sont des PNL avec des fonctions objectives linéaires sur des contraintes représentables par des cônes polyédriques et des cônes de matrices semi-définies positives. Il s'agit de sous-types de MINLP, tous deux dotés d'oracles de séparation en temps polynomial. En revanche, la catégorie plus large des problèmes NLP non convexes est généralement considérée comme intraitable [203]. En outre, le MINLP, en tant que super-type englobant le MILP, contient une multitude de problèmes insolubles.

Une grande partie de la recherche dans le domaine des MINLP tourne autour d'un principe apparemment évident.

Theorem 1.5. *Si un problème MINLP présente un ensemble réalisable compact et possède un oracle de séparation en temps polynomial pour la coque convexe de cet ensemble, alors il est traitable.*

Proof. Une preuve formelle peut être trouvée plus loin dans Lemma 1.9. □

Pour quelques problèmes MINLP, les coques convexes de leurs ensembles réalisables sont bien définies et accompagnées d'oracles de séparation en temps polynomial. Par conséquent, ces problèmes MINLP spécifiques sont traçables, ce qui implique que nous pouvons "résoudre" ces problèmes en un temps raisonnable. Le théorème ci-dessus motive les chercheurs à explorer les coques convexes de divers ensembles structurés apparaissant dans les applications.

Néanmoins, pour une grande partie des problèmes MINLP, la construction des coques convexes de leurs ensembles réalisables est une tâche redoutable. La réalisation de cette tâche impliquerait que de nombreux problèmes $\mathcal{NP}$ difficiles sont, de manière inattendue, traçables. Par conséquent, la recherche actuelle est centrée sur l'identification et l'exploitation d'approximations extérieures pratiques pour ces ensembles.

Bien que la résolution de ces problèmes "approchés" ne résolve pas directement les problèmes originaux, ils peuvent servir de tremplin précieux pour résoudre le problème original à l'aide de l'algorithme détaillé dans la section suivante. Cette caractéristique souligne la nature profonde de la recherche en cours dans le domaine des MINLP.

1.2 Optimisation MINLP basée sur la relaxation

La résolution des problèmes MINLP non convexes pose des défis importants. Néanmoins, il existe une classe notable de problèmes MINLP non convexes pour lesquels un algorithme d'énumération implicite reste utile, étant donné qu'ils remplissent la condition suivante.

Definition 1.6. *Un problème MINLP* (1.2) *est dit borné si, pour tout* $j \in [n]$, $-\infty < \ell_j < u_j < +\infty$.

Pour un problème MINLP dont l'ensemble réalisable est compact, il existe toujours un hypercube qui contient son ensemble réalisable. Dans la suite, nous considérons les problèmes MINLP à contrainte de boîte.

L'algorithme sBB (spatial Branch-and-Bound), un algorithme d'énumération implicite, sert de cadre fondamental à de nombreux solveurs MINLP d'usage général. En général, cet algorithme implique trois procédures fondamentales : Le "primal bounding", le "dual bounding" et le "branching".

Tout au long de l'exécution de l'algorithme, il garde la trace de deux bornes critiques : la borne duale et la borne primale. La meilleure solution découverte au cours de l'exécution de l'algorithme, souvent appelée solution en place, est utilisée pour établir la borne primale. L'objectif principal de la borne primale est de trouver une solution réalisable au problème MINLP.

L'algorithme sBB parcourt implicitement l'espace de recherche défini par la contrainte de la boîte du problème MINLP. Ce processus implique la subdivision de l'espace de recherche en régions plus petites et le traitement de sous-problèmes MINLP contraints dans ces sous-espaces de recherche. L'algorithme sBB se branche non seulement sur des variables entières comme l'algorithme BB classique pour les MILP, mais aussi sur des variables continues dans des expressions non linéaires et non convexes. Ce dernier comportement est appelé *branchement spatial*, qui permet des approximations plus fines des expressions non linéaires dans des régions plus petites.

La borne duale locale est une borne inférieure pour la valeur objective de toute solution dans un sous-problème MINLP contraint particulier. Si la borne primale tombe en dessous de la borne duale locale pour une région donnée, cela signifie qu'aucune solution meilleure que la solution en place ne peut exister dans cette région peu prometteuse, ce qui permet d'élaguer la recherche dans cette région.

Toutes les régions non élaguées restantes constituent collectivement l'espace de recherche "ouvert" que l'algorithme sBB doit explorer pour assurer sa convergence. La borne duale retenue par l'algorithme sBB représente la plus petite des bornes duales locales dans toutes ces régions non élaguées. L'écart de dualité, qui est la différence entre la borne primaire et la borne duale, sert à certifier la convergence de l'algorithme sBB.

Pour naviguer méthodiquement dans la région de recherche, une stratégie d'énumération systématique est employée, appelée *règle de branchement*. L'objectif d'une règle de branchement diffère : il peut s'agir de trouver une bonne solution primaire ou de réduire l'écart.

Afin d'explorer méthodiquement la région de recherche, l'algorithme sBB utilise une approche d'énumération systématique connue sous le nom de *règle de branchement*. L'objectif d'une règle de branchement peut varier ; elle peut viser à découvrir une solution primale prometteuse ou à réduire l'écart entre les bornes.

En raison de la nature modulaire de l'algorithme sBB, les composantes de la délimitation primale, de la délimitation duale et des règles de branchement peuvent être examinées indépendamment et intégrées de manière transparente, ce qui s'apparente à la philosophie de conception du solveur SCIP [62]. Les discussions approfondies sur les règles primales de bornage et de branchement sortent du cadre de cette thèse, et les lecteurs sont invités à consulter [49] et [58] pour des aperçus détaillés.

Comme nous le montrerons, de nombreuses techniques de bornage dual reposent sur la notion de "relaxations". C'est pourquoi nous insistons fortement sur le fait que l'algorithme sBB est un "algorithme MINLP basé sur la relaxation". Dans la section suivante, nous donnons des définitions formelles des relaxations.

Definition 1.7. *Etant donné un problème MINLP, sa relaxation est un autre problème MINLP, qui contient toutes les solutions réalisables du problème MINLP original.*

La définition ci-dessus est générale, et nous montrerons des méthodes concrètes pour construire des relaxations dans le chapitre suivant. Nous examinons tout d'abord les conséquences des relaxations. Normalement, un problème relaxé devrait être traitable, éventuellement sous la forme d'un problème d'optimisation convexe, ou au moins, il devrait être plus abordable en termes de calcul que le problème original. Une approche illustrative implique une stratégie géométrique, dans laquelle une approximation extérieure de l'ensemble réalisable du problème original est construite, résultant en un problème relaxé. La valeur optimale de ce problème relaxé est appelée "valeur de relaxation optimale", et la meilleure solution pour le problème relaxé est appelée "solution de relaxation optimale". De cette manière, l'algorithme sBB dérive une borne duale locale, comme indiqué ci-dessous.

Lemma 1.8. *La valeur optimale de la relaxation est au maximum la valeur optimale du problème original.*

Proof. Ceci est dû au fait que l'ensemble réalisable du problème de relaxation inclut celui du problème original. □

L'observation suivante est simple mais fondamentale pour l'optimisation non convexe.

Lemma 1.9. *Pour un ensemble compact $K \subseteq \mathbb{R}^n$, l'optimisation linéaire sur K est équivalente à l'optimisation linéaire sur* $\mathrm{conv}(K)$*.*

Proof. Il est évident que $\min_{x \in K} cx \geq \min_{x \in \mathrm{conv}(K)} cx$. De plus, pour tout $x = \sum_i$. Par conséquent, $\min_{x \in K} cx = \min_{x \in \mathrm{conv}(K)} cx$. □

Un problème de relaxation est considéré comme "étanche" lorsque sa solution optimale est également optimale pour le problème original. Par conséquent, l'obtention de l'étanchéité nécessite souvent que l'ensemble réalisable du problème relaxé corresponde à la coque convexe de l'ensemble réalisable du problème original. Néanmoins, la construction de la coque convexe peut s'avérer difficile. Par conséquent, il est souvent plus pratique de chercher une approximation extérieure de l'ensemble K qui trouve un équilibre entre la qualité de la relaxation et l'efficacité de la relaxation.

Tout au long de l'exécution de l'algorithme sBB, les règles primales de bornage et de branchement peuvent utiliser les informations obtenues à partir de la solution de relaxation optimale. Par exemple, les heuristiques de recherche locale peuvent commencer leur exploration à partir d'une solution de relaxation optimale, en l'utilisant comme point de départ pour guider leur recherche.

Néanmoins, des tâches spécifiques, telles que la réduction de l'écart de dualité, peuvent exiger exclusivement une limite duale locale sans nécessairement nécessiter la solution de relaxation. Par conséquent, ce concept introduit une interprétation plus limitée de la relaxation.

Definition 1.10. *Étant donné un problème MINLP, sa relaxation objective est un autre problème MINLP dont la valeur optimale est au maximum la valeur optimale du problème original.*

Avec l'introduction mentionnée ci-dessus, cette thèse s'attaque au défi de construire des relaxations pour un problème MINLP structuré ou une classe de problèmes MINLP structurés. Cela permet de trouver des approximations traçables qui peuvent aider à résoudre le problème original de manière efficace. Suite à l'analyse ci-dessus, la thèse traite du problème de la construction d'approximations extérieures convexes pour des ensembles non convexes.

1.3 Structure de la thèse

Nous organisons cette thèse comme suit. Dans Chap. 3, nous résumons les outils de relaxation de base dans la littérature. Dans Chap. 4, nous développons et introduisons quelques résultats de relaxation avancés pour les problèmes structurés. Dans les chapitres suivants, nous étudions les problèmes MINLP structurés et utilisons nos outils de relaxation pour résoudre ces problèmes. Dans Chap. 5, nous étudions la programmation signomiale et nous proposons des coupes d'intersection et des coupes d'approximation extérieure pour relaxer le problème. Dans Chap. 6, nous étudions la maximisation sous-modulaire et ses problèmes généralisés, et nous proposons des coupes d'intersection pour approximer ces problèmes. Dans Chap. 7, nous étudions le problème submodulaire d'empaquetage de bacs, nous appliquons la relaxation de Dantzig-Wolfe (DW) et le branch-and-price pour résoudre ce problème, et nous utilisons un algorithme d'approximation linéaire par morceaux adapté pour résoudre le problème de tarification. Dans Chap. 8, nous considérons le problème du flux de marchandises multiples non divisible dans les réseaux sans fil, où le codage de réseau est employé pour réduire le trafic. Nous comparons deux méthodes de linéarisation

pour les termes quadratiques booléens apparaissant dans ce problème, et nous proposons la relaxation de Dantzig Wolfe et l'algorithme branch-and-price pour résoudre le problème MILP linéarisé. Dans Chap. 9, nous étudions le problème du recouvrement continu sur les réseaux et introduisons des formulations MILP big-M pour modéliser les fonctions linéaires non convexes par morceaux. Dans Chap. 10, nous concluons cette thèse avec des perspectives sur la recherche future de MINLP.

1.4 Publications antérieures

Certaines parties de la thèse sont publiées à l'avance. Chap. 9 est basé sur un travail conjoint avec Mercedes Pelegrín qui est publié dans l'Omega International Journal of Management Science [249]. Chap. 8 est basé sur un travail conjoint avec Sonia Haddad Vanier qui est publié dans la revue Networks [313]. Chap. 7 est basé sur un travail conjoint avec Claudia D'Ambrosio, Sonia Haddad Vanier, et Emiliano Traversi qui sera publié dans EURO Journal on Computational Optimization [313]. Chap. 6 est basé sur un travail conjoint avec Leo Liberti qui fait l'objet d'une révision majeure dans Mathematical Programming Series B. Chap. 5 est basé sur un travail conjoint soumis avec Claudia D'Ambrosio, Sonia Haddad-Vanier, Leo Liberti.

Chapter 2

Introduction

The conventional practice of applied mathematics rests on three essential processes: modeling, simulation, and optimization.

Scientists typically hold the view that there exist well-defined principles governing both natural and human processes. Modeling is the field dedicated to uncovering and describing these principles through formal languages. Formal languages can be parsed by computers into executable code. In this thesis, we consider mathematical models that are described formally by known inputs, unknown inputs, and possibly with some decision criteria for the outputs.

The majority of natural and human processes, such as physical phenomena, are either not entirely within human control or prohibitively expensive to reproduce. Simulation creates environments for mimicking these processes using their mathematical models, facilitating the characterization of their properties. Contemporary computers enable the creation of virtual environment by translating formal language models into executable codes, subsequently executing them to produce output data. Nowadays, with powerful computing resources, simulation can now handle mathematical models with a vast number of inputs, such as simulating quantum systems and multi-agent systems.

Optimization is the mathematical process of finding values of the unknown outputs of a model that satisfies given criteria by means of observed inputs. In this setting, the mathematical model is called an *optimization model*, the problem is called an *optimization problem*. The procedure for optimization is typically referred as an *algorithm*, which can be implemented on computers. Optimization plays a crucial role in model fitting and decision-making, like the traveling salesman problem, where good outputs correspond to routes with low cost.

Though both simulation and optimization utilize computers for executing mathematical models, their objectives can diverge. Simulation aims to replicate natural and human processes based on their mathematical models, whereas optimization operates under the assumption that the provided mathematical models are always valid, focusing on the task of searching for optimal solutions.

Mathematical Programming (MP) is a specific formal language for describing most optimization problems. Every formal sentence is called a *formulation* of some given optimization problem. Every MP formulation decomposes an optimization problem (or problem class) into five symbolic entities, called: (i) parameters, which encode the problem input; (ii) decision variables, which encode the problem output; (iii) one or more objective functions, expressed in

terms of parameters and decision variables, which encode the criteria to be optimized; (iv) zero or more explicit constraints, which are criteria depending on parameters and decision variables; (v) zero or more implicit constraints, which are criteria that state membership of some decision variables in a given set. Functions and explicit constraints are explicitly given in terms of mathematical expressions, whereas the set appearing in implicit constraints must be taken into account by a solution algorithm deployed on the formulation.

Optimization under uncertainty is the investigation of optimization problems when there is uncertainty regarding the parameters involved. In the general case, addressing this uncertainty involves the incorporation of sampling methods into an optimization process, allowing for a probabilistic approximation of optimization problems through sampling. However, it is worth noting that this thesis will not delve into such matters.

Solution algorithms for MP formulations are called *solvers*. Such solvers correspond to a taxonomy of MP formulation classes organized about continuity or integrality of the decision variables, as well as linearity, nonlinearity, convexity of the objective function(s) and explicit constraints. For example, MP formulations with integer variables and linear forms are called Mixed-Integer Linear Programming (MILP). When MP formulations include both integer variables and nonlinear terms, they fall into the category known as Mixed-Integer Nonlinear Programming (MINLP).

Global optimization is the study of optimization problems involving nonlinearity and non-convexity. This is a larger class than MINLP because it also includes the so-called *black-box* optimization problems, where functions are given as oracles, rather than as mathematical expressions — but this thesis will not treat such problems.

As we will demonstrate later, MINLP can express many types of optimization problems. At present, MINLP solvers [48, 62, 283] can theoretically tackle several types of structured MINLP problems globally, including box-constrained problems described by elementary functions (such as power and trigonometric functions) and elementary convex cones (including polyhedral cones and second-order cones). However, it is crucial to note that there are subclasses of MINLP problems that can be proven to be $\mathcal{NP}$-complete or even undecidable. The topics of the complexity and computability of MINLP, as discussed in [203], fall outside the scope of this thesis.

MINLP draws upon certain methodologies from MILP, enabling it to handle problems with discrete decision variables from operations research and combinatorial optimization. MINLP goes a step further and extends its capabilities to tackle nonlinear mathematical models as well. As a result, MINLP finds relevance in diverse fields such as chemical research [253], process engineering [192], and control theory [172]. However, the current state-of-the-art limits the speed of solvers, leaving significant room for further research.

This thesis studies various approaches for solving multiple classes of MINLP problems. The main objective of this thesis is to narrow the disparity between the limited range of available optimization algorithms and the vast challenging MINLP problems. The central theme underpinning our study revolves around relaxation methods for MINLP. In the subsequent sections, we will define and categorize MINLP problems, while also providing an overview of fundamental relaxation-based optimization techniques.

2.1 Classifications of MINLP problems

A MINLP problem formally admits the following form:

$$\begin{aligned} & \inf && f_0(x) && \text{(2.1a)} \\ \text{s.t.} \quad & i \in [m] && f_i(x) \in \mathcal{S}_i && \text{(2.1b)} \\ & j \in [n] && x_j \in [\ell_j, u_j] && \text{(2.1c)} \\ & j \in [k] && x_j \in \mathbb{Z}, && \text{(2.1d)} \end{aligned}$$

where $[n] := \{1, \cdots, n\}$ is the index set of all variables, $[k] := \{1, \cdots, k\}$ is the index set of integer variables, $[m] := \{1, \cdots, m\}$ is the index set of function-in-set constraints. The objective function f_0 maps variables to a scalar value. For all $i \in [m]$, $\mathcal{S}_i$ is a set embedded in a linear space, and f_i maps variables to a vector in that linear space. For all $j \in [n]$, the scalar constants ℓ_j, u_j are lower and upper bounds of variable x_j respectively, and $-\infty \leq \ell_j < u_j \leq +\infty$. The above parameters are also called *data* of the MINLP problem.

Each constraint $x_j \in \mathbb{Z}$ in (2.1d) is called an *integrality constraint*, each constraint $x_j \in [\ell_j, u_j]$ in (2.1b) is called a *variable bound constraint*, and each constraint $f_i(x) \in \mathcal{S}_i$ is called a *function-in-set constraint*. A function-in-set constraint is composed of a function and a set, and it allows for the modeling of complicated constraints. For example, $\mathcal{S}_i$ can be a nonconvex set or a manifold. Even though it is possible to represent integrality and variable bound constraints as unique function-in-set constraints, the convention is to express them individually.

The initial step in solving MINLP problems involves identifying their types, a crucial procedure applied in general-purpose MINLP solvers. The tractability of MINLP problems relies on their specific types, which we define based on the types of variables and constraints present within them.

MINLP types are compositions of elementary types. To indicate that type T is a subtype of type T', we use the notation T $\preceq$ T'. Especially, in many programming languages, we use $\varnothing$ to denote the NULL type for syntax completeness. The NULL type is the sub-type of any type. Now, we proceed to define the elementary types and their subtypes.

The elementary type TI has sub-types in {integer, binary, $\varnothing$}, and it pertains to the discrete aspect of a MINLP problem. If there is at least one integer variable ($k \neq 0$), then TI is of the sub-type “integer”. If additionally, the variable bounds of all integer variables lie within the range $[0, 1]$, then TI is of the sub-type "binary." In the absence of any integer variables ($k = 0$), TI is represented as $\varnothing$ (NULL type). Importantly, we observe that the “binary” type is a subtype of the “integer” type (binary $\preceq$ integer).

The elementary type TC has sub-types in {linear, conic, nonlinear}, and it relates to the continuous aspect of a MINLP problem. If every function f_i is affine and its associated set S_i is a nonnegative/nonpositive orthant or zero cone, then TC is of the sub-type “linear”. If every function f_i is affine and its associated set S_i is a convex cone, then TC is of the sub-type "conic." However, if at least one function f_i is nonlinear or if at least one set $\mathcal{S}_i$ is non-polyhedral, then TC is of the sub-type “nonlinear”. It is essential to recognize that the "linear" type is a subtype of the “conic” type (linear $\preceq$ conic), and the “conic” type is a subtype of the “nonlinear” type (conic $\preceq$ nonlinear).

The conic type has some subtypes, which correspond to convex cones, such as polyhedral cone, power cone, second-order cone, cone of semi-definite matrices, and cone of composite

matrices. The nonlinear type also has some subtypes not in conic type, such as polynomial and signomial.

The elementary type TM has sub-types in {mixed, $\varnothing$}, and it captures the interplay of discrete and continuous properties. When there are some, but not all ($0 \neq k \neq n$), integer variables present in the MINLP problem, then TM is of the sub-type "mixed". However, if either all ($k = n$) or none ($k = 0$) of the variables are integer, then TM is of the sub-type $\varnothing$ (NULL). It is worth noting that $\varnothing \preceq$ mixed, meaning a continuous problem is a subtype of a mixed problem.

Definition 2.1. *A type of MINLP is a product type of TM,TI,TC.*

In a MINLP problem of elementary types TM, TI, TC, its continuous relaxation becomes a nonlinear programming (NLP) problem of a type of TC. Thus, each MINLP type has a corresponding derived type for its continuous relaxation.

Definition 2.2. *A derived type TD of MINLP type TM, TI, TC belongs to {convex, nonconvex}. If any problem instance of the TC type is convex, then the derived type TD of TC is convex; otherwise, TD is nonconvex.*

Typically, MINLP problems are referred to as "TM TI TC programs", denoting their elementary types, and discrete and continuous properties. Sometimes, we also examine the derived type TD, in which case we call it a "TD TM TI TC program". To simplify matters, we frequently employ abbreviations based on a straightforward rule: we preserve and capitalize the initial letters of TM, TI, TC, and "program," while TD remains unabbreviated.

As an illustration, consider the following examples: If a MINLP problem is of the type "mixed, integer, linear", it is referred to as a "mixed-integer linear program" and is abbreviated as MILP. If the MINLP problem is a MILP and additionally, its TI is binary, it becomes a "mixed binary linear program" and is abbreviated as MBLP. If the type of a MINLP problem is "integer, polynomial", then its derived type is nonconvex. In this case, it is known as an "integer polynomial program" (IPP) or a "nonconvex integer polynomial program" (nonconvex IPP).

In the following alternative representation, the MINLP problem can be viewed as a linear optimization problem over its feasible set. To transform the formulation (2.1) into this representation, the following processes can be taken: add the constraint $f(x) \leq t$, where t is an additional variable; change the objective of the optimization problem to $\inf t$; consider t as a part of the variables in the problem. This gives rise to the alternative formulation of MINLP problems:

$$\begin{aligned} & \inf && cx && (2.2a)\\ \text{s.t.} \quad & i \in [m] && f_i(x) \in \mathcal{S}_i && (2.2b)\\ & j \in [n] && x_j \in [\ell_j, u_j] && (2.2c)\\ & j \in [k] && x_j \in \mathbb{Z}. && (2.2d) \end{aligned}$$

Without loss of generality, we focus on MINLP problems represented in the formulation of (2.2). A MINLP problem is considered *tractable* if there exists an algorithm capable of approximating it with arbitrary accuracy in a time that is polynomial w.r.t. its encoding size and the desired level of accuracy. A vast number of tractable MINLP problems are (continuous)

convex optimization problems, and the tractability can be achieved by the ellipsoid algorithm [132]. This algorithm relies on the following concept of separation oracle.

Definition 2.3. *Given a compact convex set $K \subseteq \mathbb{R}^n$, a separation oracle for K is an oracle (black box) that, given a vector $x \in \mathbb{R}^n$, returns one of the following:*

- *Assert that $x \in K$.*
- *Find a hyperplane that separates x from K: a vector $a \in \mathbb{R}^n$, such that $ay > ax$ for all $y \in K$.*

Since we consider the MINLP problem as in (2.2), we refer to the separation from MINLP as the separation from its feasible set. The classification of MINLP problems helps identify the first subclass of tractable MINLP problems.

Lemma 2.4 ([293])**.** *If a convex NLP problem with a compact feasible set has a polynomial time separation oracle, then it is tractable.*

Linear programming (LP) and semi-definite programming (SDP) are NLP with linear objective functions over constraints represetable by polyhedral cones and cones of positive semi-definite matrices. They are well-established and tractable sub-types of MINLP, both equipped with polynomial time separation oracles. In contrast, the broader category of nonconvex NLP problems is typically considered intractable [203]. Furthermore, MINLP, as a super-type encompassing MILP, contains a multitude of intractable problems.

Much of the research in the field of MINLP revolves around a seemingly self-evident principle.

Theorem 2.5. *If a MINLP problem exhibits a compact feasible set and possesses a polynomial time separation oracle for the convex hull of this set, then it is tractable.*

Proof. A formal proof can be found later in Lemma 2.9. □

For a few MINLP problems, the convex hulls of their feasible sets are well-defined and accompanied by polynomial time separation oracles. As a result, these specific MINLP problems are tractable, implying that we can "solve" these problems in a reasonable time. The theorem above motivates researchers to explore the convex hulls of various structured sets arising in applications.

Nevertheless, for a substantial portion of MINLP problems, constructing the convex hulls of their feasible sets is a formidable task. Achieving this would imply that numerous $\mathcal{NP}$-hard problems are, unexpectedly, tractable. Consequently, the ongoing research direction is centered on identifying and harnessing practical outer approximations for these sets.

While solving these "approximated" problems may not directly resolve the original problems, they can serve as valuable stepping stones towards solving the original problem using the algorithm detailed in the subsequent section. This characteristic underscores the profound nature of ongoing research in the field of MINLP.

2.2 Relaxation-based MINLP optimization

Solving nonconvex MINLP problems poses significant challenges. Nevertheless, there is a notable class of nonconvex MINLP problems for which an implicit enumeration algorithm remains useful, given they meet the following condition.

Definition 2.6. *A MINLP* (2.2) *is said to be box-bounded, if, for all* $j \in [n]$, $-\infty < \ell_j < u_j < +\infty$.

For a MINLP problem with a compact feasible set, there always exists a hypercube, which contains its feasible set. W.l.o.g., we consider box-constrained MINLP problems in the sequel.

The spatial Branch-and-Bound (sBB) algorithm [204], an implicit enumeration algorithm, serves as a foundational framework for numerous general-purpose MINLP solvers. Generally, this algorithm involves three fundamental procedures: "primal bounding", "dual bounding", and "branching".

Throughout the algorithm's execution, it keeps track of two critical bounds: the *dual bound* and the *primal bound*. The best solution discovered during the algorithm's run, often referred to as the incumbent solution, is utilized to establish the primal bound. The primary objective of the primal bounding is to find a feasible solution to the MINLP problem.

The sBB algorithm implicitly traverses the search space defined by the box constraint of the MINLP problem. This process involves subdividing the search space into smaller regions and addressing constrained MINLP subproblems within these sub-search spaces. The sBB algorithm not only branches on integer variables like the classical BB algorithm for MILP but also branches on continuous variables in nonlinear and nonconvex expressions. The latter behavior is called *spatial branching*, which allows for finer approximations of nonlinear expressions within smaller regions.

The local dual bound is a lower bound for the objective value of any solution within a particular constrained MINLP subproblem. If the primal bound falls below the local dual bound for a given region, it signifies that no solution better than the incumbent solution can exist within that unpromising region, allowing for pruning the search there.

All remaining unpruned regions collectively constitute the "open" search space that the sBB algorithm needs to explore for its convergence. The dual bound retained by the sBB algorithm represents the smallest among the local dual bounds within all these unpruned regions. The *(duality) gap*, which is the difference between the primal bound and the dual bound, serves as a certification of the sBB algorithm's convergence.

To methodically navigate through the search region, a systematic enumeration strategy is employed, referred to as a *branching rule*. The goal of a branching rule differs: it can be either to find a good primal solution or to reduce the gap.

In order to methodically explore the search region, the sBB algorithm employs a systematic enumeration approach known as a *branching rule*. The purpose of a branching rule can vary; it may aim to discover a promising primal solution or to narrow the gap between bounds.

Due to the modular nature of the sBB algorithm, the components of primal bounding, dual bounding, and branching rules can be examined independently and seamlessly integrated, akin to the design philosophy of the SCIP solver [62]. In-depth discussions of primal bounding and branching rules are outside the scope of this thesis, and readers are referred to [49] and [58] for comprehensive insights.

As will be shown, many dual bounding techniques hinge on the notion of "relaxations". Therefore, we strongly emphasize the sBB algorithm as a "relaxation-based MINLP algorithm". In the following section, we provide formal definitions of relaxations.

Definition 2.7. *Given a MINLP problem, its relaxation is another MINLP problem, which contains all feasible solutions to the original MINLP problem.*

The above definition is general, and we will show concrete methods to construct relaxations in the next chapter. We first look at the consequence of relaxations. Normally, a relaxed problem should be tractable, possibly taking the form of a convex optimization problem, or at the very least, it should be more computationally affordable than the original problem. One illustrative approach involves a geometric strategy, wherein an outer approximation of the feasible set of the original problem is constructed, resulting in a relaxed problem. The optimal value of this relaxed problem is designated as the "optimal relaxation value", and the best solution for the relaxed problem is referred to as the "optimal relaxation solution". In this manner, the sBB algorithm derives a local dual bound, as below.

Lemma 2.8. *The optimal relaxation value is at most the optimal value of the original problem.*

Proof. This is because the feasible set of the relaxation problem includes that of the original problem. □

The following observation is simple but fundamental for nonconvex optimization.

Lemma 2.9. *For a compact set $K \subseteq \mathbb{R}^n$, linear optimization on K is equivalent to linear optimization on $\operatorname{conv}(K)$.*

Proof. It is obvious that $\min_{x\in K} cx \geq \min_{x\in \operatorname{conv}(K)} cx$. Moreover, for every $x = \sum_i \lambda_i x^i \in \operatorname{conv}(K)$ with $x_i \in K, \lambda_i \geq 0$, since $\sum_i \lambda_i = 1$, $cx = c(\sum_i \lambda_i x^i) \leq \min_{x\in K} cx$. Therefore, $\min_{x\in K} cx = \min_{x\in \operatorname{conv}(K)} cx$. □

A relaxation problem is considered "tight" when its optimal solution is also optimal to the original problem. Consequently, achieving tightness frequently necessitates that the feasible set of the relaxed problem corresponds to the convex hull of the original problem's feasible set. Nevertheless, constructing the convex hull can be challenging. Therefore, it is often more practical to seek an outer approximation of set K that strikes a suitable balance between the quality of the relaxation and the efficiency of the relaxation.

Throughout the execution of the sBB algorithm, both primal bounding and branching rules can make use of the insights gleaned from the optimal relaxation solution. For example, local search heuristics may commence their exploration from an optimal relaxation solution, using it as a starting point to guide their search.

Nonetheless, specific tasks, such as decreasing the duality gap, might exclusively demand a local dual bound without necessarily requiring the relaxation solution. Consequently, this concept introduces a more limited interpretation of relaxation.

Definition 2.10. *Given a MINLP problem, its objective relaxation is another MINLP problem whose optimal value is at most the optimal value of the original problem.*

With the aforementioned introduction, this thesis tackles the challenge of constructing relaxations for a structured MINLP problem or a class of structured MINLP problems. This helps find tractable approximations that can aid in solving the original problem effectively. Following the above analysis, the thesis deals with the problem of constructing convex outer approximations for nonconvex sets.

2.3 Structure of the thesis

We organize this thesis as follows. In Chap. 3, we summarize the basic relaxation tools in the literature. In Chap. 4, we develop and introduce some advanced relaxation results for structured problems. In the following chapters, we study structured MINLP problems and use our relaxation tools to solve these problems. In Chap. 5, we study signomial programming, and we propose intersection cuts and outer approximation cuts to relax the problem. In Chap. 6, we study submodular maximization and its generalized problems, and we propose intersection cuts for approximating these problems. In Chap. 7, we study the submodular bin packing problem, we apply Dantzig-Wolfe (DW) relaxation and branch-and-price to solve this problem, and we use a tailored piece-wise linear approximation algorithm to solve the pricing problem. In Chap. 8, we consider the problem of unsplittable multi-commodity flow in wireless networks, where network coding is employed to reduce traffic. We compare two linearization methods for Boolean quadratic terms arising in this problem, and we propose Dantzig Wolfe relaxation and branch-and-price algorithm to solve the linearized MILP problem. In Chap. 9, we study the problem of continuous covering on networks, and introduce big-M MILP formulations for modeling nonconvex piece-wise linear functions. In Chap. 10, we conclude this thesis with the perspectives on the future research of MINLP.

2.4 Prior publications

Parts of the thesis are published in advance. Chap. 9 is based on a joint work with Mercedes Pelegrín that is published in the Omega International Journal of Management Science [249]. Chap. 8 is based on a joint work with Sonia Haddad Vanier that is published in the Networks journal [313]. Chap. 7 is based on a joint work with Claudia D'Ambrosio, Sonia Haddad Vanier, and Emiliano Traversi that will be published in EURO Journal on Computational Optimization [313]. Chap. 6 is based on a joint work with Leo Liberti that is under a major revision in Mathematical Programming Series B. Chap. 5 is based on a submitted joint work with Claudia D'Ambrosio, Sonia Haddad-Vanier, Leo Liberti.

Chapter 3

Theory: basic relaxation methods

This chapter offers an overview of the core relaxation methods utilized in this thesis. While there is a wide variety of MINLP problem types, there are established relaxation tools readily available in the existing literature. Consequently, when tackling a MINLP problem, one can explore these readily accessible tools and select the most appropriate one to meet their specific requirements. These relaxation tools can be categorized as follows: relaxations via lifting, relaxations via submodularity, relaxations via piece-wise linear functions, and relaxations tightening via intersection cuts.

3.1 Relaxations via lifting

In this section, we present a meta-relaxation method known as "lifting". Lifting involves the approximation of a set by representing it in a higher-dimensional space, thereby providing greater flexibility in addressing challenging problems. In this section, we present several lifting-based relaxation methods: Dantzig-Wolfe relaxations, factorable programming relaxations, and certificate-based relaxations. We also present the projection method that can project a high-dimensional sets into a lower dimensional space.

We have established that any MINLP problem can be reformulated into a linear optimization problem over its feasible set, which may be nonconvex. The MINLP problem's feasible set is denoted as $K \in \mathbb{R}^n$, and we approach the MINLP problem in the form of $\min_{x \in K} cx$. To construct the extended formulation of the MINLP problem, we adopt a set-theoretic approach that relies on representing the nonconvex set K in a higher-dimensional space through lifting.

Definition 3.1. *For the nonconvex set $K \in \mathbb{R}^n$, its convex hull admits a lifted representation $K' \in \mathbb{R}^{n+k}$ for $k \geq 1$, if $\mathrm{conv}(K) = \mathrm{proj}_{\mathbb{R}^n}(K')$.*

In some cases, lifting simplifies the process of constructing the convex hull for the nonconvex feasible set K. Let $c' = (c, 0)$ where $0 \in \mathbb{R}^k$. We have the following equalities:

$$\min_{x \in K} cx = \min_{x \in \mathrm{conv}(K)} cx = \min_{y \in K'} c'y. \tag{3.1}$$

As y is in the extended space containing x, we refer to $\min_{y \in K'} c'y$ as an extended formulation of the MINLP problem. In situations where constructing K' is not feasible, we instead search for a convex set $\bar{K'}$ that includes K'. The convex outer approximation $\bar{K'}$

remains valuable as it provides a relaxation $\min_{y\in\bar{K}'} c'y$ in the extended space. We term this relaxation an *extended relaxation*, which satisfies that

$$\min_{y\in K'} c'y \geq \min_{y\in\bar{K}'} c'y. \tag{3.2}$$

The lifting method offers extended formulations or extended relaxations for the MINLP problem. We have the option to solve the extended formulations/relaxations directly or improve the original projected formulations/relaxations by incorporating the results from lifting.

3.1.1 Dantzig-Wolfe relaxation

The first lifting method utilizes a geometric approach based on Dantzig-Wolfe (DW) relaxation [154, 296], which proves to be widely applicable whenever all extreme points of a nonconvex set can be enumerated. The method of the DW relaxation involves an implicit generation of convex combination of those extreme points, and, in some cases, to obtain a relaxation, the method may not exhaust all the extreme points.

Consider $K = K_1 \cap K_2$. We assume that computing the convex hulls of both K_1 and K_2 is straightforward, and we also have a lifted representation, denoted as K_2', such that $\mathrm{conv}(K_2) = \mathrm{proj}_{\mathbb{R}^n}(K_2')$. With these assumptions in place, we can derive a convex outer approximation $\bar{K}$ using the following procedure.

Lemma 3.2. $K \subseteq \bar{K} := \mathrm{conv}(K_1) \cap \mathrm{proj}_{\mathbb{R}^n}(K_2')$.

Proof. The convex hull of the intersection of two sets is included in the intersection of the convex hulls of the two sets. □

In the following, we consider exclusively the sets with polytope convex hulls, for which one can generate all their extreme points in a finite time. It is noteworthy that a convex set may have an infinite number of extreme points, and we could in principle extend our methods for such case.

It is important to note that a polytope can have two different representations: the hyperplane representation and the vertex representation. In the case where $\mathrm{conv}(K_2)$ is a polytope and its vertices V_2 are known, we can examine its vertex representation. An explicit lifted representation of $\mathrm{conv}(K_2)$ can be defined as follows:

$$K_2' = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}_+^{V_2} : \sum_{v\in V_2} y_v = 1 \wedge\ x = \sum_{v\in V_2} y_v v\}, \tag{3.3}$$

where $k = |V_2|$. We note that $\mathrm{conv}(K_2) = \mathrm{conv}(V_2) = \mathrm{proj}_{\mathbb{R}^n}(K_2')$.

By Lemma 3.2, the problem $\min_{x\in\bar{K}} cx$ is a convex relaxation of $\min_{x\in K} cx$. As $\bar{K} = \mathrm{conv}(K_1) \cap \mathrm{proj}_{\mathbb{R}^n}(K_2')$, we call the relaxation *DW relaxation*. In addition, the DW relaxation

admits the following simplified form:

$$\begin{aligned} \mathrm{DW}(V) := \min \quad & c \sum_{v \in V} y_v v && \text{(3.4a)} \\ \text{s.t.} \quad & \sum_{v \in V} y_v v \in \mathrm{conv}(K_1) && \text{(3.4b)} \\ & \sum_{v \in V} y_v = 1 && \text{(3.4c)} \\ & y \in \mathbb{R}_+^V, && \text{(3.4d)} \end{aligned}$$

where $V = V_2$ and x is substituted by $\sum_{v \in V_2} y_v v$. However, the number of vertices V_2 can be exponential in n, and this limits the tractability of the DW relaxation.

The column generation method [37, 296] is employed to address this issue. It begins by considering a subset V_2' of the vertices V_2 and then solves the restricted problem $\mathrm{DW}(V_2')$. Next, it searches for a point (column) $v \in V_2 \setminus V_2'$ that can improve the restricted problem and adds this column to V_2'. This process iterates, resulting in a sequence of non-increasing upper bounds $\mathrm{DW}(V_2')$. Finally, the column generation process stops generating new columns once the bound converges, i.e., $\mathrm{DW}(V_2) = \mathrm{DW}(V_2')$.

The column generation process should determine whether and how to generate a column. We assume that $\mathrm{conv}(K_1)$ is a polytope with a known hyperplane representation, i.e., $\mathrm{conv}(K_1) = \{x \in \mathbb{R}^n : \forall i \in [m], a^i x \leq b_i\}$, where $a_i \in \mathbb{R}^n, b_i \in \mathbb{R}$. Consequently, the DW relaxation can be expressed as an LP:

$$\begin{aligned} \mathrm{DW}(V) := \min \quad & c \sum_{v \in V} y_v v && \text{(3.5a)} \\ \text{s.t. } \forall i \in [m] \quad & a^i \sum_{v \in V} y_v v \leq b_i && \text{(3.5b)} \\ & \sum_{v \in V} y_v = 1 && \text{(3.5c)} \\ & y \in \mathbb{R}_+^V. && \text{(3.5d)} \end{aligned}$$

Since the strong duality holds for LP, $\mathrm{DW}(V)$ equals the dual optimal value of the dual LP:

$$\begin{aligned} \mathrm{DW}(V) := \max \quad & \lambda - \sum_{i \in [m]} b_i \mu_i && \text{(3.6a)} \\ \text{s.t. } \forall v \in V \quad & \sum_{i \in [m]} a^i v \mu_i + cv - \lambda \leq 0 && \text{(3.6b)} \\ & \mu \in \mathbb{R}_+^m, \lambda \in \mathbb{R}. && \text{(3.6c)} \end{aligned}$$

The duality gives rise to a certificate for the optimality of $\mathrm{DW}(V_2')$ and a verifiable condition to decide whether to generate a column.

Lemma 3.3. *Let μ', λ' be the dual optimal solution to the dual problem for $\mathrm{DW}(V_2')$. If for all $v \in V_2 \setminus V_2'$, $\sum_{i \in [m]} a^i v \mu_i + cv - \lambda' \leq 0$, then μ' is also a dual optimal solution to the dual problem for $\mathrm{DW}(V_2)$.*

Proof. This condition implies that the dual solution is also feasible for the dual problem associated with $\mathrm{DW}(V_2)$, as the primal problem is already feasible. Thus, the primal value equals the dual value. By the strong duality of LP, the primal-dual pairs are both optimal. □

Hence, if every constraint in V_2 is met by μ', then $\mathrm{DW}(V_2)$ equals $\mathrm{DW}(V_2')$. At this point, the column generation process can be halted, and we can obtain the primal solution for $\mathrm{DW}(V_2')$, which concurrently serves as a primal optimal solution for $\mathrm{DW}(V_2)$. To verify the "all-satisfied" condition, it is sufficient to solve the following *pricing subproblem*:

$$\max_{v \in V} \sum_{i \in [m]} a^i \mu_i' v + cv \tag{3.7}$$

and compare the maximum with λ'. If the maximum value of the pricing subproblem is strictly less than λ', then the corresponding maximum argument is added to V_2'. However, if the maximum value is equal to or greater than λ', it indicates that the primal and dual problems have converged, and the DW relaxation is considered solved.

We next consider a more concrete case, where $V = \{0,1\}^n \cap P$, and P is a polytope given in hyper-plane representation. The pricing subproblem thus admits the following MILP representation:

$$\max_{x \in P \cap \{0,1\}^n} (\sum_{i \in [m]} a^i \mu_i')x + cx, \tag{3.8}$$

which can be solved by a MILP solver. In a more general setting, P can be a convex set instead of a polyhedron.

3.1.2 Projection

A topic closely related to extended formulations is projections. In some cases, an explicit formulation of $\mathrm{conv}(K)$ is only known in the extended space, but it is more efficient and convenient to work in the original projected space. The *projection approach* seeks a low-dimensional approximation of $\mathrm{proj}_{\mathbb{R}^n}(K')$.

In the following two cases, it becomes impractical to store the complete descriptions of $\mathrm{conv}(K)$ or K'. Firstly, when given K' as a polytope, the number of facets of $\mathrm{proj}_{\mathbb{R}^n}(K')$ may be exponential in that of K'. Secondly, if K' is in vertex representation as K_2', the number of auxiliary variables y in its lifted representation can become very large. Consequently, this further hinders the storage of the entire representation.

To address these practical challenges, a cutting plane algorithm is used to iteratively refine a convex outer approximation of K through projections. The algorithm operates like the column generation method for the dual LP.

Let us illustrate the usage of projection with an example. Vertex polyhedrality is a useful property for convexifying nonconvex functions in MINLP. A specific case is when a function is convex-extensible from vertices.

Definition 3.4. *Let X be a polytope, and let Q be the vertices of X. A function $f : X \to \mathbb{R}$ is convex-extensible from vertices, if*

$$\operatorname{conv}((x,t) \in X \times \mathbb{R} : f(x) \le t) = \\ \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \exists y \in \mathbb{R}_+^Q \ \sum_{q \in Q} y_q = 1, x = \sum_{q \in Q} y_q q, \sum_{q \in Q} y_q f(q) \le t\}. \quad (3.9)$$

A convex function g is a *convex underestimating function* of f over X, if for all $x \in X$, $g(x) \le f(x)$. The *convex envelope* F_f is defined as the maximal convex underestimating function of f over X. Hence, the convex envelope F_f of a convex-extensible function f is entirely determined by its values at vertices Q. The epigraph K' of the convex envelope F_f possesses a lifted representation that is similar to that of K_2':

$$K' := \{(x,y,t) \in \mathbb{R}^n \times \mathbb{R}_+^Q \times \mathbb{R} : \sum_{q \in Q} y_q = 1, x = \sum_{q \in Q} y_q q, \sum_{q \in Q} y_q f(q) \le t\}. \quad (3.10)$$

Let $(\tilde{x}, \tilde{t})$ be a point to be separated, where we can let $\tilde{t}$ be $f(\tilde{x})$ or other values. The projection problem asks a cutting plane $(a, 1)$ to separate $(\tilde{x}, \tilde{t})$ from $\operatorname{proj}_{\mathbb{R}^{n+1}}(K')$. This separation problem can be formulated as the following LP:

$$\begin{aligned} & \max && a\tilde{x} + \tilde{t} && (3.11a) \\ \text{s.t. } & \forall q \in Q && aq + f(q) \le 0 && (3.11b) \\ & && (a,1) \in C, && (3.11c) \end{aligned}$$

where C is a convex set imposing the boundedness of $(a, 1)$. In practice, C is defined by a bound constraint on L_1 or L_2 norm of $(a, 1)$, which is LP representable. A successful separation returns $(a, 1)$ such that $a\tilde{x} + \tilde{t} > 0$.

Lemma 3.5 ([135])**.** *Every concave function is convex-extensible from vertices.*

Concave functions are not as tractable as convex functions, however, we can construct the convex envelopes of concave functions using the above results.

3.1.3 Factorable programming

We next present a second relaxation approach that is widely adopted by MINLP optimization solvers. This approach relies on the symbolic representation of a mathematical program. The syntax and semantics of the symbolic representation can be defined formally. For brevity, we here only give a high-level introduction, and we refer to [201, 204] for more details.

General nonconvex NLP problems typically admit the following formulation:

$$\min_{x \in \mathbb{R}^n} \ c \cdot x \quad \text{s.t.} \quad Ax + Bg(x) \le d, \quad (3.12)$$

where $c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{m \times k}, g : \mathbb{R}^n \to \mathbb{R}^k, d \in \mathbb{R}^m$. The map $g(x)$ represents a vector $(g_1(x), \ldots, g_k(x))$ of nonconvex functions on x, and we refer to g_i as its *terms*. This formulation can be converted from the formulation (2.2) through epigraphical reformulation.

The backend convex relaxation algorithms implemented in many general-purpose solvers, including `BARON`, `Couenne`, and SCIP, are convex relaxations. Most of them further convert the

convex relaxations into LP relaxations. These solvers leverage the separability present in the rows of $Ax + Bg(x)$, allowing them to relax and linearize nonlinear terms g_i individually.

In the solvers' data structures, the problem (3.12) is transformed into an extended formulation:

$$\min_{(x,y)\in\mathbb{R}^{n+k}} c \cdot x \quad \text{s.t.} \quad Ax + By \leq d \ \wedge \ y = g(x). \tag{3.13}$$

All the nonlinear terms are grouped within the nonconvex constraints $y = g(x)$. These constraints give rise to a nonconvex *lifted set* defined as:

$$\mathcal{S}_{\text{lift}} \coloneqq \{(x, y) \in \mathbb{R}^{n+k} : y = g(x)\}. \tag{3.14}$$

In fact, one may find the above lifted structure similar to the DW relaxation.

The relaxation algorithms employed by these solvers are based on factorable programming [201, 223]: this approach treats the multivariate nonlinear terms g_i as composite functions. These algorithms commonly factor each g_i into sums and products of a collection of univariate functions. If convex and concave relaxations of those univariate functions are available, these algorithms can linearize these relaxations, and yield a linear relaxation for Eq. (3.12). Common lists of such univariate functions, that are usually available to all sBB solvers, include t^a (for $a \in \mathbb{N}$), $\frac{1}{t}$, $\log t$, $\exp t$. Some solvers also offer a choice of trigonometric functions, e.g. `Couenne`. In this way, one can obtain a convex outer approximation of $\mathcal{S}_{\text{lift}}$, which yields a convex relaxation of the NLP.

3.1.4 Certificate-based relaxation

The third lifting method is an algebraic method based on non-negativity certificates, and we call the relaxations derived from this method. It proves to be particularly useful for polynomial programming (PP) and related problems. An advantage of this method is that it does not necessitate box constraints, which are essential for many conventional relaxation methods. This relaxation method is based on the duality point of view.

Assume that we aim to solve the following problem:

$$\lambda^* := \min_{x\in K} f(x), \tag{3.15}$$

where K represents a complicated domain of the nonlinear function f. The problem has an equivalent dual formulation, which searches for the maximum $\lambda \in \mathbb{R}$ such that $f(x) - \lambda$ is non-negative over K:

$$\lambda^* := \max\{\lambda \in \mathbb{R} : \forall x \in K \ f(x) - \lambda \geq 0\}, \tag{3.16}$$

We assume that $f_\lambda(x) := f(x) - \lambda$ belongs to a set F of functions, such as polynomials. Let F_K^+ denote the set of non-negative functions in F over K. Each function in F_K^+ is referred to as a *non-negativity certificate*. Additionally, we assume that F_K^+ forms a convex cone, such as the cone of nonnegative polynomials. As a result, the dual problem (3.16) becomes a convex optimization problem:

$$\max\{\lambda \in \mathbb{R} : f_\lambda \in F_K^+\}. \tag{3.17}$$

As optimization over F_K^+ is often intractable, our objective is to approximate F_K^+. To achieve this, we aim to construct nested families of conic inner approximations of F_K^+ indexed by level numbers i: $F_K^1 \subseteq \cdots F_K^i \cdots \subseteq F_K^m = F_K^+$, where the maximum level number m can approach infinity. Thus, each family F_K^i at level i results in a restriction of the dual problem:

$$\lambda^i := \max\{\lambda \in \mathbb{R} : f_\lambda \in F_K^i\}. \tag{3.18}$$

It follows that λ^i is a lower bound of λ^* and is non-decreasing.

Lemma 3.6. $\lambda^1 \leq \cdots \leq \lambda^m = \lambda^*$.

Proof. The results follow from $F_K^1 \subseteq \cdots F_K^i \cdots \subseteq F_K^m = F_K^+$. □

Let us provide a geometric interpretation of (3.18). This interpretation allows us to extract a primal solution after solving (3.18).

We consider that $f - \lambda$ and f belong to a linear space consisting of a specific class of functions, such as the space of polynomials. In this linear space, we have a basis denoted as $\{r^t(x)\}_{t\in[T]}$, which could be, for example, a set of monomials. Consequently, $f(x)$ can be expressed as a linear combination of these basis functions: $f(x) = \sum_{t\in[T]} f_t r^t(x)$.

We take F_K^i as a subset of the linear space, and its elements are parameterized by coefficients of the basis functions. Let $Y := \{y \in \mathbb{R}^T : \exists x \in K, \forall t \in [T], y_t = r^t(x)\}$ represent a lifted representation of the basis functions $\{r^t(x)\}_{t\in[T]}$.

Lemma 3.7. *For all $i \in [m]$, $Y \subseteq (F_K^i)^*$, where $(F_K^i)^*$ is the dual cone of F_K^i.*

Proof. $Y \subseteq (F_K^i)^*$ if and only if for every $y \in Y$, $gy \geq 0$ holds for every $g \in F_K^i$. This is true, since there exists an $x \in K$ such that $gy = \sum_{t\in[T]} g_t y_t = \sum_{t\in[T]} g_t r^t(x)$. □

Consequently, the dual cone $(F_K^i)^*$ forms a convex outer approximation of the lifted set Y. We denote the coefficient vector of $f(x) := \sum_{t\in[T]} f_t r^t(x)$ as f, and we take indifferently between f and $f(x)$. Then the optimal value of the dual problem (3.18) for the primal problem (3.15) is equal to:

$$\lambda^i := \min\{fy : y \in (F_K^i)^*\}. \tag{3.19}$$

Since $Y \subseteq (F_K^i)^*$, we call (3.19) the level-i (primal) relaxation. Sometimes, we also call (3.18) the level-i (dual) relaxation. The duality pairs the primal cone F_K^i and the dual cone $(F_K^i)^*$. This pairing also establishes the duality between the level-i primal relaxation (3.19) and the level-i dual relaxation (3.18).

For each dual relaxation solution, there exists a corresponding function $f_{\lambda^i} \in F_K^i$, which is paired with a vector y in the dual cone. As a result, one can extract an approximated lifted representation from y and deduce a primal solution x.

The lifting method has two different interpretations in the primal and the dual sense. The primal interpretation is straightforward: $(F_K^i)^*$ is a lifted convex outer approximation of the nonconvex set Y. For example, $f_{\lambda'}$ is a polynomial, then y is outer approximations of the monomials of $f_{\lambda'}$. The dual interpretation reveals that F_K^i are inner approximations F_K^+.

We look at a binary polynomial programming (BPP) example, where $K = \{0,1\}^n$ and the polynomial f has degree $d \leq n$.

We review two types of non-negativity certificates over K. The first certificate is the sum of squares (SOS) polynomial [197], for whcih

$$F_K^i := \{g(x) : g(x) := \sum_i g_i(x), \text{ s.t. } \forall i \; g_i(x) := h_i^2(x)\}, \tag{3.20}$$

where h_i are polynomials with bounded degrees. The resulting relaxation is called Lasserre relaxation and can be solved via SDP.

The second certificate is the sum of bound-factor product (SOBFP) polynomial [196, 197]:

$$F_K^i := \{g(x) : g(x) := \sum_i g_i(x), \text{ s.t. } g_i(x) := \sum_{S_i, S_i' \subseteq [n] : S_i \cap S_i' = \varnothing} \prod_{j \in S_i} x_j \prod_{j \in S_i'} (1 - x_j)\}. \tag{3.21}$$

The resulting relaxation is called Sherali-Adams relaxation [276] and can be solved via LP.

These relaxations certify a lower bound λ of f over K by finding a sum g of non-negativity certificates g_i such that $g = f - \lambda$. It is possible that the degrees of the polynomials g_i are larger than the degree d of f. However, the monomials of degrees higher than d sum to zero in g, i.e., they are "canceled out". Therefore, the lifting method can decompose $f - \lambda$ into polynomials of higher degrees. This redundancy imply that the complexity of certifying non-negativity increases with respect to the degrees of the certificates.

3.2 Relaxations via submodularity

We have demonstrated that constructing tight relaxations for MINLP problems often involves finding the convex hull of certain nonconvex sets. In this section, we show that the concept of submodularity can help find convex relaxations for certain discrete functions.

Submodular functions are important models of discrete convex functions. The classical definition of submodular set functions [212] is equivalent to the definition of submodular functions over the Boolean hypercube through Boolean indicator-characterization of subsets. The latter definition is used in this thesis:

Definition 3.8. *A function $f : \{0,1\}^n \to \mathbb{R}$ is called a submodular function, if for every $x, y \in \{0,1\}^n$, $f(x) + f(y) \geq f(\max(x,y)) + f(\min(x,y))$, where $\min, \max$ are element-wise minimum and maximum.*

This definition can be generalized over any Cartesian product of subsets of $\mathbb{R}$ [287]. The work of Jack Edmonds [133] plays a prominent role in the study of the combinatorial properties of submodular functions. We refer to [270] for basic concepts and definitions. The convex envelope of a submodular function f is its Lovász extension [23, 212]. The framework of convex analysis can be adapted to discrete settings, and discrete convex funcitons are a generalization of submodular functions. We refer to [234] for more details about discrete convex analysis.

We can further define other discrete functions based on the submodularity.

Definition 3.9. *A function is supermodular if its negative is submodular. A modular function is both submodular and supermodular. A submodular-supermodular (SS) function is the difference between two submodular functions.*

Thereby, affine functions are modular. The Fenchel conjugate of a continuous (possibly nonconvex) function is a function that encodes its convex envelope. A min-max theorem (namely, Fenchel duality) holds for any continuous function and its Fenchel conjugate [177]. Convexity is a desirable property, as computing Fenchel conjugates of many convex functions, such as convex quadratics, is tractable. In contrast, submodular functions are discrete functions. Nevertheless, it is possible to derive a discrete generalization of the Fenchel conjugate as follows.

Definition 3.10 ([143])**.** *Given a discrete function $g : \{0,1\}^n \to \mathbb{R}$, its Fenchel conjugate $g^\star : \mathbb{R}^n \to \mathbb{R}$ is defined as $g^\star(y) := \max_{x\in\{0,1\}^n}(xy - g(x))$.*

Submodular functions [234] have the following discrete generalization of the Fenchel duality.

Lemma 3.11 ([143])**.** *Given a submodular function $g : \{0,1\}^n \to \mathbb{R}$, its Fenchel conjugate $g^\star$ is convex, and $\min_{x\in\{0,1\}^n} g(x) = \max_{y\in\mathbb{R}^n}(-g^\star(y))$.*

We consider the following set of f:

$$K^c := \{(x,t) \in \{0,1\}^n \times \mathbb{R} : cf(x) \le t\}, \tag{3.22}$$

where $c \in \{-1, 1\}$.

Our goal is to create a convex outer approximation of K^c. To achieve this, we construct the convex envelope of f when $c = 1$, and we generate a concave overestimator of f when $c = -1$. These two constructions yield a convex outer approximation of K^c. Notably, when $c = 1$, the convex outer approximation becomes tight, resulting in the best convex outer approximation of K^c.

3.2.1 Convex envelope

The convex envelope of f over $[0,1]^n$ is called its Lovász extension. The construction of Lovász extension relates the facets of the convex envelope to several combinatorial structures defined as follows.

Recall that a permutation σ on $[n]$ is a bijective map from $[n]$ to itself. The map $\sigma(i) \in [n]$ is the image of an element $i \in [n]$ under this permutation. We denote by S_n the set of permutations on $[n]$. We define the following sets and vectors related to permutations.

Definition 3.12. *Given a permutation $\sigma \in S_n$ and an integer $i \in \{0, \dots, n\}$, define $\sigma([i]) := \{\sigma(1), \dots, \sigma(i)\}$ ($\sigma([0]) := \varnothing$), and define $v^i(\sigma) := \sum_{j\in\sigma([i])} 1_j$, where 1_j is the j-th element vector in $\mathbb{R}^n$.*

The *convex envelope* F_f is defined as the maximal convex underestimating function of f over $\mathcal{B}$. We can then construct the convex envelope of f.

Theorem 3.13 ([23])**.** *Define the map $a_f : S_n \to \mathbb{R}^n$ such that it satisfies $a_f(\sigma)_{\sigma(i)} = f(v^i(\sigma)) - f(v^{i-1}(\sigma))$ for all $\sigma \in S_n$ and $i \in [n]$. Then $\mathsf{F}_f(x) := \max_{\sigma\in S_n} a_f(\sigma)x$ is the convex envelope of f over $[0,1]^n$.*

Thus, $a_f(\sigma)x$ ($\sigma \in S_n$) a *facet* of the convex envelope F_f. Thm. 3.13 shows that permutations on $[n]$ are in one-to-one correspondence to the facets of F_f. Moreover, the convex envelope F_f is a piece-wise linear function.

We next look at the relation between facets and permutations.

Corollary 3.14. *Given a permutation $\sigma \in S_n$, for all $i \in [n] \cup \{0\}$, $a_f(\sigma)x$ defines a facet that $a_f(\sigma)v^i(\sigma) = f(v^i(\sigma))$.*

Proof.

$$a_f(\sigma)v^i(\sigma) = \sum_{j\in[i]} a_f(\sigma)_{\sigma(j)} = \sum_{j\in[i]} \Big(f(v^j(\sigma)) - f(v^{j-1}(\sigma))\Big) = f^i(v^i(\sigma)) - f(0) = f(v^i(\sigma)),$$

where the first equation follows from Defn. 3.12, the second equation follows from Lemma 4.12, and the last two equations follow from the expansion of the sum. □

Conversely to Cor. 3.14, given a point in $\{0,1\}^n$, we can construct all the facets equal to f at it.

Corollary 3.15. *For a point $v \in \{0,1\}^n$, let ι be the number of ones in v. If a permutation $\sigma \in S_n$ satisfies that $v = v^\iota(\sigma)$, then the facet $a_f(\sigma)x$ admits that $a_f(\sigma)v = f(v)$.*

Given $\tilde{x} \in [0,1]^n$, the value of the convex envelope $\mathsf{F}_f(\tilde{x})$ equals

$$\max_{\sigma\in S_n} a_f(\sigma)\tilde{x}. \tag{3.23}$$

Moreover, an optimal solution σ^* defines a facet $\sigma(\sigma^*)x$ such that $\sigma(\sigma^*)\tilde{x} = \mathsf{F}_f(\tilde{x})$. Therefore, the argument of the evaluation problem is also a solution to the facet separation problem. A strongly polynomial time *sorting algorithm* can solve the evaluation problem [23]: Let $\sigma^* \in S_n$ be a permutation such that $\tilde{x}_{\sigma^*(1)} \geq \cdots \geq \tilde{x}_{\sigma^*(n)}$, then an optimal solution to (3.23) is $\sigma(\sigma^*)$.

3.2.2 Concave overestimator

We next construct a concave overestimator for f, which is also a piece-wise linear function. The facets of the concave overestimator are defined as follows:

Theorem 3.16 ([237])**.** *For every $x' \in \{0,1\}^n$, the following affine functions overestimates f:*

$$\begin{aligned} f^1_{x'}(x) :=& f(x') - \sum_{j\in[n]:x'_j=1} (f(1) - f(1-1_j))\,(1-x_j) + \sum_{j\in[n]:x'_j=0} (f(x'+1_j) - f(x'))\,x_j, \\ f^2_{x'}(x) :=& f(x') - \sum_{j\in[n]:x'_j=1} (f(x') - f(x'-1_j))\,(1-x_j) + \sum_{j\in[n]:x'_j=0} (f(1_j) - f(0))\,x_j, \end{aligned}$$

where 1_j is the j-th unit vector, and 1 is the all-one vector.

From the above affine overestimators, we construct the piece-wise linear overesimator:

$$\bar{f}(x) := \max_{x'\in\{0,1\}^n, i\in\{1,2\}} f^i_{x'}(x). \tag{3.24}$$

However, we are not aware of a polynomial-time algorithm to separate a facet of $\bar{f}$. In [237], the overesimator has the same values as f over the Boolean hypercube.

3.3 Relaxations via piece-wise linear approximations

Previous methods for constructing convex relaxations primarily involve generating convex outer approximations of nonconvex sets. In most cases, the feasible set of the MINLP problem is an intersection of multiple nonconvex sets, where each set corresponds to a nonconvex constraint.

In such cases, conventional convex relaxations may fail to be exact when each nonconvex constraint is convexified individually. Assume that the feasible set $K = K_1 \cap K_2 \in \mathbb{R}^2$, where $K_1 := \left\{(x,y) : y \geq \begin{cases} (|x|-1)^2 & |x| \geq 1 \\ 1-x^2 & |x| \leq 1 \end{cases}\right\}$ and $K_2 := \{(x,y) : x \geq 0\}$. Let the optimization problem be $\min_{(x,y)\in K} -x-y$, and the optimal solution is $((-1+\sqrt{5})/2, (-1+\sqrt{5})/2)$. Figs. 3.1a to 3.1d shows $K_1, K_2, K, \mathrm{conv}(K_1) \cap K_2$. However, solving the relaxation over $\mathrm{conv}(K_1) \cap K_2$ gives a solution $(0,0)$. Therefore, the relaxation is not exact, and the relaxation solution is also far from the optimal solution.

Alternatively, one can utilize a nonconvex outer approximation of K_1, as long as optimization over the approximation set remains feasible. The corresponding relaxation is, therefore, nonconvex. Given the current capabilities of MILP solvers, we consider MILP relaxations in this context. The nonconvex outer approximation is commonly referred to as a *piece-wise linear* (PWL) approximation. For example, see a PWL outer approximation $\bar{K}_1$ of K_1 in Fig. 3.1e.

We will now introduce a general nonconvex relaxation method based on PWL functions. Next, we will formally define PWL outer approximations.

We recall that the convex hull of $h+1$ affinely independent points is called an h*-simplex* (simplex). To do so, we will utilize a geometric view of simplicial complexes.

Definition 3.17. *A simplicial complex C is a collection of h-simplices in $\mathbb{R}^h$, such that*

- *Any face of a $\sigma \in C$ is also in C;*
- *For all $\sigma.\tau \in C$, their intersection $\sigma \cap \tau$ is a face of each of them.*

Then we define simplicial covers.

Definition 3.18. *Given a full-dimensional nonconvex set $K \subseteq \mathbb{R}^n$, a simplicial cover of K is a collection $\{P_t\}_{t\in[T]}$ of convex polyhedrons, such that*

- $K \subseteq \cup_{t\in[T]} P_t$*;*
- *For all $t_1, t_2 \in [T], t_1 \neq t_2$, $\mathrm{int}(P_{t_1}) \cap \mathrm{int}(P_{t_2}) = \varnothing$;*
- $\{P_t\}_{t\in[T]}$ *is a simplicial complex.*

Every simplicial cover yields a PWL outer approximation of K:

$$\bar{K} := \{x \in \mathbb{R}^n : \exists t \in [T]\ x \in P_t\} \tag{3.25}$$

Assume that P_t are in hyperplane representation, such that $P_t = \{x \in \mathbb{R}^n : \forall i \in I_t\ a^{ti}x \leq b^{ti}\}$. We assume that the recession cone of P_t is zero, i.e., $\{x \in \mathbb{R}^n : \forall i \in I_t\ a^{ti}x \leq 0\} = \{0\}$. Using the disjunctive programming principle [29], we obtain a MILP representation of $\bar{K}$:

$$\bar{K} = \{x : \exists y \in \{0,1\}^T\ z \in \mathbb{R}^{Tn}, x = \sum_{t\in[T]} z^t, 1 = \sum_{t\in[T]} y^t, \forall i \in I_t, a^{ti}z^t \leq b^{ti}y^t\} \tag{3.26}$$

Therefore, a MILP relaxation of the nonconvex optimization problem $\min_{x\in K} cx$ is:

$$\begin{aligned} \min \quad & cx && \text{(3.27a)}\\ \text{s.t. } \forall t \in T, i \in I_t \quad & a^{ti}z^t \le b^{ti}y^t && \text{(3.27b)}\\ & x = \sum_{t\in[T]} z^t && \text{(3.27c)}\\ & 1 = \sum_{t\in[T]} y^t && \text{(3.27d)}\\ & y \in \{0,1\}^T, z \in \mathbb{R}^{Tn} && \text{(3.27e)} \end{aligned}$$

We note that this representation is an extended formulation.

In high-dimensional spaces, constructing simplicial covers can be challenging, and solving the MILP (3.27) may become computationally expensive. Consequently, practical algorithms often resort to PWL relaxations for sets in dimensions $n = 1, 2$. For instance, when K is the hypograph of a convex univariate or bivariate function. In the following example, we construct a PWL approximation for the case of $n = 1$.

A PWL function is linear on each piece of a given partition of its domain. Let f be the univariate convex function over $[\underline{x}, \overline{x}]$ with $\underline{x} \ge 0$. We say a value of the variable x a *breakpoint*. Given an ordered set of breakpoints $\mathcal{B} = (x_1, x_2, \dots, x_h)$ such that $x_k \in [\underline{x}, \overline{x}]$ ($k \in [h] := \{1, \dots, h\}$), $x_1 = \underline{x}$ and $x_h = \overline{x}$, the following PWL function approximates f over the domain $[\underline{x}, \overline{x}]$:

$$\bar{f}_{\mathcal{B}}(x) := \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}(x - x_k) + f(x_k), \text{ for } x_k \le x \le x_{k+1}, 1 \le k \le h-1.$$

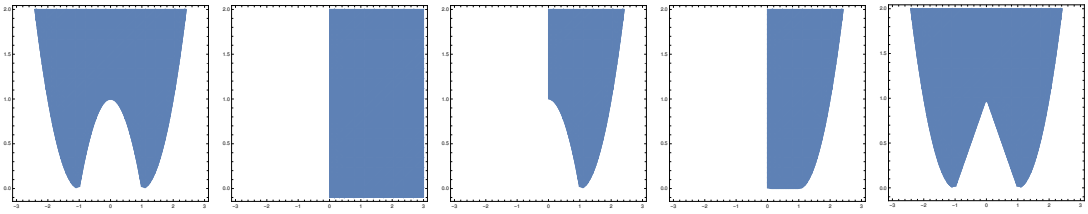
Note that $\bar{f}_{\mathcal{B}}$ is an *over-estimator* of f due to the convexity of f. We call $\bar{f}_{\mathcal{B}}$ a *PWL approximation* of f. Applying (3.25), we obtain a MILP representation of the PWL approximation of the hypograph of f:

$$\begin{aligned} &\{(x,t) \in [\underline{x}, \overline{x}] \times \mathbb{R} : f(x) \ge t\} \subseteq \{(x,t) \in [\underline{x}, \overline{x}] \times \mathbb{R} : \exists y \in \{0,1\}^{h-1} \ \wedge \ x = \sum_{k\in[h-1]} z_k \wedge \\ &1 = \sum_{k\in[h-1]} y_k \wedge \forall k \in [h-1]\, x_k y_k \le z_k \le x_{k+1} y_k \wedge \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(z_k - x_{k-1}z_k) + f(x_{k-1})y_k \le t\}. \end{aligned} \tag{3.28}$$

We call $\mathcal{B}$ a breakpoint set in $[\underline{x}, \overline{x}]$, and $\bar{f}_{\mathcal{B}}$ its induced PWL function. Note that we consider the two bounds $\underline{x}$ and $\overline{x}$ as breakpoints here. The approximation error is expressed as ℓ_p-norm of the difference between the PWL approximation and the target function.

Definition 3.19. *Given a set $\mathcal{B} \subset [\underline{x}, \overline{x}]$ of breakpoints, the ℓ_p approximation error of $\bar{f}_{\mathcal{B}}$ with respect to f over $[\underline{x}, \overline{x}]$ is defined as $\ell_p(\bar{f}_{\mathcal{B}}, f) := (\int_{\underline{x}}^{\overline{x}} |\bar{f}_{\mathcal{B}}(x) - f(x)|^p \, dw)^{\frac{1}{p}}$.*

Although adding breakpoints decreases the approximation error, it increases the computation resource to solve the PWL relaxation. So a common problem is to understand the best achievable approximation error given a fixed number of breakpoints (limited computational resource).



(a) K_1 (b) K_2 (c) K (d) $\mathrm{conv}(K_1) \cap K_2$ (e) $\bar{K}_1$

Figure 3.1 PWL approximation of nonconvex sets.

3.4 Relaxation tightening via intersection cuts

The cutting plane algorithm aims to construct a polyhedral outer approximation P of the nonconvex set $\mathcal{S}$, which is the feasible set of the MINLP problem $\min_{x \in \mathcal{S}} cx$. Thereby, the polyhedron P yields an LP relaxation of the MINLP problem. Intersection cuts are a particular type of valid inequalities that can tighten the polyhedral outer approximation.

The construction of intersection cuts [99] requires two key ingredients: a simplicial cone containing $\mathcal{S}$, and an $\mathcal{S}$-free set, which is defined as follows.

Definition 3.20. *Given a set $\mathcal{S} \subsetneq \mathbb{R}^p$, a closed set $\mathcal{C}$ is (convex) $\mathcal{S}$-free if $\mathcal{C}$ is convex and $\mathrm{int}(\mathcal{C}) \cap \mathcal{S} = \varnothing$.*

Thinking reversely, $\mathcal{S}$-free sets are convex regions whose interiors have no element of $\mathcal{S}$, so $\mathcal{S}$-free sets can describe "non-feasible" regions of a MINLP problem.

Fig. 3.3 shows an example of a $\mathcal{S}$-free set, where we find that $\mathcal{C}$ is a convex inner approximation of $\mathrm{cl}(\mathcal{S}^c)$.

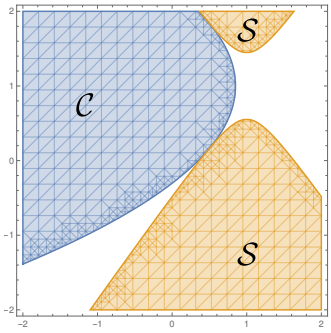
Intersection cuts were initially devised in the continuous setting (the papers [291], cited in [178, Ch. III], appeared before the classic paper [30]), where they could approximate the hypograph $\mathcal{S}$ of a convex function over a polytope. There is a unique maximal $\mathcal{S}$-free set: the epigraph of that convex function. Later, intersection cuts were used in the discrete setting [30], where $\mathcal{S}$ is a lattice. Several more families of lattice-free sets (e.g., splits, triangles, and spheres [99, 202]) were described later.

We show how to construct $\mathcal{S}$-free sets from a "reverse" representation of some nonconvex sets. We look at sets involving a particular type of nonconvex functions.

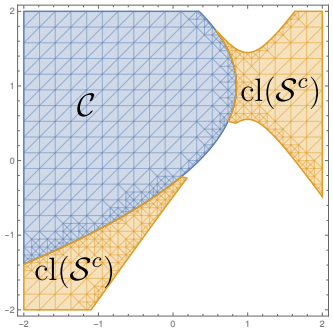
Definition 3.21. *A function f is said to be difference-of-concave (DCC), if there exists two concave functions f_1, f_2 such that $f = f_1 - f_2$.*

It is easy to show that the negative of a DCC function is also a DDC function, and thus any DC function is also a DCC function. A nonconvex set admits a *DCC formulation*, if it is represented as the sublevel set of a DCC function. We call such a set *a DCC set*. The superlevel set of a DCC function is a sublevel set of another DDC function (the negative of that function), so one can reformulate the superlevel set into a DCC set. For a function f and a point $\tilde{x}$ in its domain, we denote the first-order approxiamtion $f(\tilde{x}) + \nabla f(\tilde{x})(x - \tilde{x})$ of f as $\Xi^f_{\tilde{x}}(x)$. The following lemma gives a family of $\mathcal{S}$-free sets for DCC sets via *linearization method*.

Lemma 3.22 ([271])**.** *Let $\mathcal{S} := \{x \in \mathbb{R}^p : f_1(x) - f_2(x) \leq 0\}$, where f_1, f_2 are concave functions over $\mathbb{R}^p$. Then for any $\tilde{x} \in \mathbb{R}^p$, $\mathcal{C} := \{x \in \mathbb{R}^p : f_1(x) - \Xi^{f_2}_{\tilde{x}}(x) \geq 0\}$ is $\mathcal{S}$-free. Moreover, if $\tilde{x} \in \mathbb{R}^p \setminus \mathcal{S}$, $\tilde{x} \in \mathrm{int}(\mathcal{C})$.*



(a) $\mathcal{C}$ as an $\mathcal{S}$-free set.



(b) $\mathcal{C}$ as an inner approximation of $\mathrm{cl}(\mathcal{S}^c)$.

Figure 3.2 An $\mathcal{S}$-free set $\mathcal{C}$.

To apply the above lemma, it suffices to reverse the inequality defining $\mathcal{S}$ and linearize its convex part. We refer to $\tilde{x}$ as a *linearization point* of f_2. Note that when the common domain $\mathcal{G}$ of f_1 and f_2 is not $\mathbb{R}^p$, $\mathcal{S}$ should be restricted to the *ground set* $\mathcal{G}$.

We show some examples in Fig. 3.2.

Given an $\mathcal{S}$-free set, the next step is to construct an intersection cut. The construction procedure requires additionally a translated polyhedral cone $\mathcal{R}$ such that $\mathcal{S} \subseteq \mathcal{R}$ and the vertex $\tilde{x}$ of $\mathcal{R}$ is not in $\mathcal{S}$. Let us suppose that $\mathcal{R}$ admits a hyper-plane representation: $\{x \in \mathbb{R}^p : B(x - \tilde{x}) \leq 0\}$, where B is a $p \times p$ invertible matrix. For all $j \in [p]$, let r^j denote the j-th column of $-B^{-1}$, then r^j turns out to be an extreme ray of $\mathcal{R}$. Thereby, $\mathcal{R}$ also admits a ray representation $\{x \in \mathbb{R}^p : \exists \eta \in \mathbb{R}^p_+ \; x = \tilde{x} + \sum_{j=1}^p \eta_j r^j\}$,

For all $j \in [p]$, we define the *step length* from $\tilde{x}$ along ray r_j to the boundary $\mathrm{bd}(\mathcal{C})$ as

$$\eta_j^* := \max_{\eta_j \in [0,+\infty]} \{\eta_j : \tilde{x} + \eta_j r^j \in \mathcal{C}\}. \tag{3.29}$$

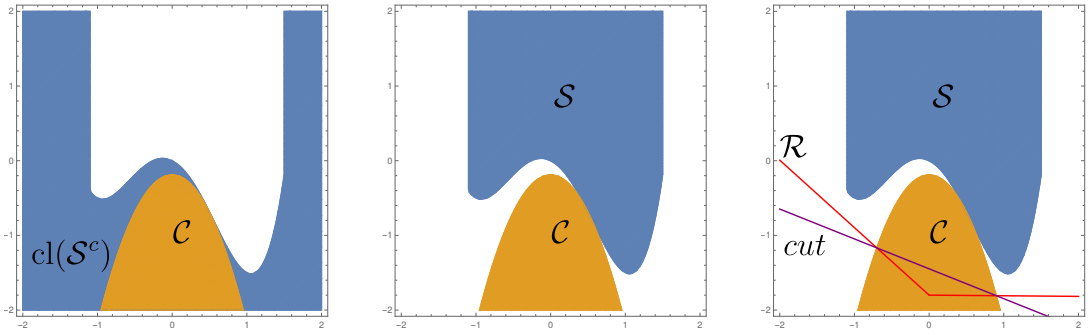
Then, the intersection cut admits the form:

$$\sum_{j=1}^{p} \frac{1}{\eta_j^*} B_j(x - \tilde{x}) \leq -1, \tag{3.30}$$

where B_j is the j-th row of B. When all step lengths are positive, the above linear inequality cuts off $\tilde{x}$ from $\mathcal{S}$. The construction of an intersection cut is visualized in Fig. 3.3.

In practice, we can obtain $\mathcal{S}$, $\mathcal{R}$, and $\tilde{x}$ as follows. Assume that we have an LP relaxation $\min_{x \in \mathcal{P}} cx$ of the MINLP problem, where $\mathcal{P}$ is a polyhedral outer approximation of the feasible set. If the LP solution is not feasible to SP, as the LP relaxation usually comprises all linear constraints of the MINLP problem, then the solution must not satisfy some nonlinear constraint. Thus, we can set $\tilde{x}$ to the LP solution and define $\mathcal{S}$ by the nonlinear constraint. Moreover, we can extract $\mathcal{R}$ from the basis of the LP defining $\tilde{x}$.

The main issue we address is therefore the construction of (maximal) $\mathcal{S}$-free sets. The reason why we look for *maximal* such sets is that, if $\mathcal{C}$ and $\mathcal{C}^*$ are two $\mathcal{S}$-free sets with $\mathcal{C} \subseteq \mathcal{C}^*$, then the intersection cut derived from $\mathcal{C}^*$ dominates the intersection cut derived from $\mathcal{C}$. Thereby, we give a formal definition of maximal $\mathcal{S}$-free sets.



(a) $\mathcal{C}$ as an inner approximation of $\mathrm{cl}(\mathcal{S}^c)$.

(b) $\mathcal{C}$ as an $\mathcal{S}$-free set.

(c) Simplicial cone $\mathcal{R}$ and the intersection cut.

Figure 3.3 An $\mathcal{S}$-free set $\mathcal{C}$, simplicial cone $\mathcal{R}$, and intersection cut.

Definition 3.23. *Given a closed convex set $\mathcal{G} \subseteq \mathbb{R}^p$ such that $\mathcal{S} \subsetneq \mathcal{G}$, an $\mathcal{S}$-free set $\mathcal{C}$ is (inclusion-wise) maximal in $\mathcal{G}$, if there is no other $\mathcal{S}$-free set $\mathcal{C}'$ such that $\mathcal{C} \cap \mathcal{G} \subsetneq \mathcal{C}' \cap \mathcal{G}$.*

Defn. 3.23 generalizes the conventional definition of maximal $\mathcal{S}$-free sets, and one can recover the conventional definition by setting $\mathcal{G} = \mathbb{R}^p$. In some cases, it is difficult to study the maximality of $\mathcal{S}$-free sets in $\mathbb{R}^p$. Defn. 3.23 allows us to study intersections of $\mathcal{S}$-free sets with the *ground set* $\mathcal{G}$.

3.5 Conclusion

In this section, we present a class of common relaxation methods. In the next chapter, we introduce some advanced theoretical results for relaxing structured sets. They yield new relaxation methods. In the rest of the thesis, we apply these methods to tackle applications that can be modelled as MINLP problems.

Chapter 4

Theory: advanced structural results

Within this chapter, we introduce innovative theoretical findings related to the process of convexification or relaxation of structured sets. These findings have been developed to address practical challenges encountered throughout this thesis and will serve as the foundation for the development of cutting-edge algorithms for various problems. These results can be regarded as advanced concepts building upon the submodularity and intersection cut framework introduced in Chap. 3.

4.1 $\mathcal{S}$-free sets for structured sets

In this section, we introduce advanced results concerning the intersection cut framework. We have demonstrated that the intersection cuts relies on the concept of $\mathcal{S}$-free sets, which in turn, depend on the specific problem structure. The set $\mathcal{S}$ under consideration encompasses sets originating from NLP and those arising in submodular optimization.

4.1.1 Maximal $\mathcal{S}$-free sets for lifted sets

We consider the extended formulation (3.13) of a general NLP problem and focus on the associated lifted set $\mathcal{S}_{\text{lift}}$ in (3.14). We show a lifting result on the construction of maximal $\mathcal{S}_{\text{lift}}$-free sets.

Let $z \coloneqq (x, y)$ denote the vector variable in the extended formulation (3.13), with its index set being $[n + k]$. Consequently, we have $z_{[n]} = x$ and $z_{[n+1:n+k]} = y$. Consider a closed subset $\mathcal{X}$ of the domain $\bigcap_{i\in[k]} \operatorname{dom}(g_i)$ for x, and let $\mathcal{Y}$ be a closed subset of the domain $\bigtimes_{i\in[k]} \operatorname{range}(g_i)$ for y. The ground set $\mathcal{G}$, where the variables actually vary, can be set as $\mathcal{X} \times \mathcal{Y}$. Consequently, the lifted set $\mathcal{S}_{\text{lift}}$ in (3.14) admits the form $\{(x, y) \in \mathcal{G} : y = g(x)\}$.

Given that each $g_i(x)$ (for $i \in [k]$) may only depend on a subset of variables indexed by $J_i \subseteq [n]$, we can express $g_i(x)$ as a lower order function $g'_i(x_{J_i})$ defined over $\mathbb{R}^{|J_i|}$. Let $I_i \coloneqq J_i \cup \{i + n\}$. As above, we consider a closed subset $\mathcal{X}^i$ of $\operatorname{dom}(g'_i)$ and $\mathcal{Y}^i$ of $\operatorname{range}(g'_i)$. Consequently, the graph, epigraph, and hypograph of g'_i reside within sets $\mathcal{G}^i \coloneqq \mathcal{X}^i \times \mathcal{Y}^i$, e.g., $\operatorname{epi}(g'_i) = \{(x_{J_i}, y_i) \in \mathcal{G}^i : g'_i(x_{J_i}) \leq y_i\}$.

We refer to $\mathcal{X}, \mathcal{Y}, \{\mathcal{X}^i, \mathcal{Y}^i\}_{i\in[k]}$ as the *underlying sets* of the lifted set $\mathcal{S}_{\text{lift}}$. The sets are said to be *1d-convex decomposable* by a collection $\{\mathcal{D}_j\}_{j\in[n+k]}$ of closed convex sets in $\mathbb{R}$, if $\mathcal{X} = \bigtimes_{j\in[n]} \mathcal{D}_j, \mathcal{Y} = \bigtimes_{j\in[n+1:n+k]} \mathcal{D}_j$, and, for all $i \in [k]$, $\mathcal{X}^i = \bigtimes_{j\in J_i} \mathcal{D}_j, \mathcal{Y}^i = \mathcal{D}_{n+i}$. This decomposability condition restricts the domains to Cartesian products of real lines, intervals, or half rays, thereby excluding complicated domain structures.

The decomposability condition allows for the analysis of sets involving fewer variables. Constructing $\text{epi}(g_i')$-free sets and $\text{hypo}(g_i')$-free sets is generally easier than constructing $\mathcal{S}_{\text{lift}}$-free sets. We show that every maximal $\text{epi}(g_i')$-free or $\text{hypo}(g_i')$-free set can be lifted into a maximal $\mathcal{S}_{\text{lift}}$-free set.

Theorem 4.1. *Suppose the underlying sets of $\mathcal{S}_{\text{lift}}$ are 1d-convex decomposable and g is continuous. For some $i \in [k]$, let $\mathcal{C}$ be a maximal $\text{epi}(g_i')$-free set or a maximal $\text{hypo}(g_i')$-free set in $\mathcal{G}^i$. Then $\bar{\mathcal{C}} \coloneqq \mathcal{C} \times \mathbb{R}^{|I_i^c|}$ ($I_i^c = [n+k] \setminus I_i$) is a maximal $\mathcal{S}_{\text{lift}}$-free set in $\mathcal{G}$.*

Proof. It suffices to consider the case that $\mathcal{C}$ is a maximal $\text{epi}(g_i')$-free set in $\mathcal{G}^i$. W.l.o.g., we can assume that $\mathcal{C}, \mathcal{G}^i$ are full-dimensional in $\mathbb{R}^{|I_i|}$. Since $\text{epi}(g_i')$ includes $\text{gr}(g_i')$, $\mathcal{C}$, as an $\text{epi}(g_i')$-free set, is also $\text{gr}(g_i')$-free. First, we prove that $\mathcal{C}$ is a maximal $\text{gr}(g_i')$-free set in $\mathcal{G}^i$. Assume, to aim at a contradiction, that $\mathcal{C}'$ is a $\text{gr}(g_i')$-free set that $\mathcal{C} \cap \mathcal{G}^i \subsetneq \mathcal{C}' \cap \mathcal{G}^i$. Suppose that $\text{epi}(g_i') \cap \text{int}(\mathcal{C}' \cap \mathcal{G}^i)$ is not empty and contains (x'_{J_i}, y_i'). As $\mathcal{C}$ is $\text{epi}(g_i')$-free, there exists a point $(x_{J_i}, y_i) \in \text{int}(\mathcal{C} \cap \mathcal{G}^i) \subseteq \text{int}(\mathcal{C}' \cap \mathcal{G}^i)$ such that $(x_{J_i}, y_i) \in \text{hypo}(g_i')$. It follows from the continuity of g_i' that there exists a point $(x^*_{J_i}, y_i^*) \in \text{gr}(g_i')$ in the line segment joining (x_{J_i}, y_i) and (x'_{J_i}, y_i'). As $\text{int}(\mathcal{C}' \cap \mathcal{G}^i)$ is convex, we have that $(x^*_{J_i}, y_i^*) \in \text{int}(\mathcal{C}' \cap \mathcal{G}^i)$, which leads to a contradiction to $\text{gr}(g_i')$-freeness of $\mathcal{C}'$. Therefore, $\text{epi}(g_i') \cap \text{int}(\mathcal{C}' \cap \mathcal{G}^i)$ must be empty, so $\mathcal{C}' \cap \mathcal{G}^i \subseteq \text{hypo}(g_i')$. This means that $\mathcal{C}'$ is also $\text{epi}(g_i')$-free. However, note that $\mathcal{C} \cap \mathcal{G}^i \subsetneq \mathcal{C}' \cap \mathcal{G}^i$, this contradicts with the fact that $\mathcal{C}$ is a maximal $\text{epi}(g_i')$-free set in $\mathcal{G}^i$. Therefore, $\mathcal{C}$ is a maximal $\text{gr}(g_i')$-free set in $\mathcal{G}^i$. Secondly, we prove that $\bar{\mathcal{C}}$ is a maximal $\mathcal{S}_{\text{lift}}$-free set in $\mathcal{G}$. Assume, to aim at a contradiction, that there exists an $\mathcal{S}_{\text{lift}}$-free set $\bar{\mathcal{D}}$ in $\mathcal{G}$ such that $\bar{\mathcal{C}} \cap \mathcal{G} \subsetneq \bar{\mathcal{D}} \cap \mathcal{G}$. We look at their orthogonal projections on $\mathbb{R}^{|I_i|}$. It follows from the decomposability that $\mathcal{C} \cap \mathcal{G}^i = \mathcal{C} \cap \text{proj}_{\mathbb{R}^{|I_i|}}(\mathcal{G}) = \text{proj}_{\mathbb{R}^{|I_i|}}(\bar{\mathcal{C}} \cap \mathcal{G}) \subseteq \text{proj}_{\mathbb{R}^{|I_i|}}(\bar{\mathcal{D}} \cap \mathcal{G})$. Denote $\mathcal{D} \coloneqq \text{cl}(\text{proj}_{\mathbb{R}^{|I_i|}}(\bar{\mathcal{D}} \cap \mathcal{G}))$, which is a closed convex set in $\mathcal{G}^i$. Since $\bar{\mathcal{C}} = \mathcal{C} \times \mathbb{R}^{|I_i^c|}$, $\mathcal{D}$ must strictly include $\mathcal{C} \cap \mathcal{G}^i$. Note that $\mathcal{D}$ is $\text{gr}(g_i')$-free. Since $\mathcal{C}$ is a maximal $\text{gr}(g_i')$-free set in $\mathcal{G}^i$, this implies that $\mathcal{C} \cap \mathcal{G}^i = \mathcal{D}$, which leads to a contradiction. □

For any $i \in [k]$, we call the operation $\mathcal{C} \times \mathbb{R}^{|I_i^c|}$ the *orthogonal lifting* of $\mathcal{C}$ w.r.t. g_i. A similar lifting result for integer programming is provided by Lemma 4.1 of [102]: given $\mathcal{S} \coloneqq \mathbb{Z}^n \times \mathbb{R}^h$, any maximal lattice-free set (i.e., $\mathbb{Z}^n$-free set) can be transformed into a maximal $\mathcal{S}$-free set through orthogonal lifting. Therefore, Thm. 4.1 serves as the NLP counterpart to that lemma (whose proof is also similar). This theorem allows us to focus on low-dimensional projections of the lifted set.

We will show in Cor. 5.2 that the signomial lift satisfies the prerequisites of Thm. 4.1. The following example illustrates the application of Thm. 4.1.

Example 4.2. *Consider a lifted set $\mathcal{S}_{\text{lift}}$ defined as*

$$\{(x_1, x_2, x_3, x_4, y_1, y_2, y_3) : y_1 = \exp(x_1 - x_2/x_3), y_2 = \log(x_1), y_3 = \sin(x_1/x_4)\}.$$

One can verify that the 1d-convex decomposable condition holds for $\mathcal{D}_1 = \mathbb{R}_+$, $\mathcal{D}_j = \mathbb{R}$ (for $j \in [2:7]$). Then $\mathcal{G} \coloneqq \mathbb{R}^1_+ \times \mathbb{R}^6$. We use $\log(x_1)$ to construct a $\mathcal{S}_{\text{lift}}$-free set. A maximal

$\mathcal{S}_{\text{lift}}$-free set can be $\{(x_1, x_2, x_3, x_4, y_1, y_2, y_3) \in \mathcal{G} : y_2 \leq \log(x_1)\}$. Since $\log(x_1)$ is defined over positive reals, this example gives a reason to restrict maximality over $\mathcal{G}$.

4.1.2 Maximal $\mathcal{S}$-free for DCC constraints

We provide sufficient conditions for the maximality of $\mathcal{S}$-free sets for two general classes of non-convex sets $\mathcal{S}$. To begin with, we review some fundamental results from convex analysis. Our subsequent presentation relies on the use of support functions of convex sets. The properties of support functions can be summarized as follows.

Lemma 4.3 (Chapter C of [177])**.** *For a full-dimensional closed convex set $\mathcal{C} \subsetneq \mathbb{R}^p$, let $\sigma_{\mathcal{C}} : \mathbb{R}^p \to \mathbb{R}, \lambda \mapsto \sup_{z \in \mathcal{C}} \lambda \cdot z$ be the support function of $\mathcal{C}$. Then: (i) $\mathcal{C} = \{z \in \mathbb{R}^p : \lambda \cdot z \leq \sigma_{\mathcal{C}}(\lambda), \forall \lambda \in \text{dom}(\sigma_{\mathcal{C}})\}$, (ii) $\text{int}(\mathcal{C}) = \{z \in \mathbb{R}^p : \lambda \cdot z < \sigma_{\mathcal{C}}(\lambda), \forall \lambda \in \text{dom}(\sigma_{\mathcal{C}}) \setminus \{0\}\}$, (iii) $\sigma_{\mathcal{C}}(\rho\lambda) = \rho\sigma_{\mathcal{C}}(\lambda)$ for any $\rho > 0$. Moreover, for any closed convex set $\mathcal{C}'$ including $\mathcal{C}$, $\sigma_{\mathcal{C}} \leq \sigma_{\mathcal{C}'}$.*

A valid inequality $a \cdot z \leq b$ of $\mathcal{C}$ is called a *supported valid inequality*, if there exists a *supporting point* $z' \in \text{bd}(C)$ such that $a \cdot z' = b$. Geometrically, a closed convex set is the intersection of half-spaces associated with supported valid inequalities.

Observation 4.4. *It follows from Lemma 4.3 that every supported valid inequality of $\mathcal{C}$ must admit the form $\lambda \cdot z \leq \sigma_{\mathcal{C}}(\lambda)$ for some $\lambda \in \text{dom}(\sigma_{\mathcal{C}})$, where the supremum $\sigma_{\mathcal{C}}(\lambda)$ is attained at its supporting points.*

An inequality of the form $\lambda \cdot z \leq \sigma_{\mathcal{C}}(\lambda)$, for $\lambda \in \text{dom}(\sigma_{\mathcal{C}})$, is referred to as an *exposed valid inequality*, if there exists an *exposing point* $z' \in \text{bd}(C)$ such that $\lambda \cdot z' = \sigma_{\mathcal{C}}(\lambda)$ and for all $\lambda' \in \text{dom}(\sigma_{\mathcal{C}}) \setminus \{\rho\lambda\}_{\rho > 0}$, $\lambda' \cdot z' < \sigma_{\mathcal{C}}(\lambda')$.

Observation 4.5. *An exposed valid inequality must be a supported valid inequality. Conversely, a supported valid inequality is an exposed valid inequality, if manifold $\text{bd}(\mathcal{C})$ is smooth at its supporting point. For example, $\mathcal{C}_1 := \{(x, y) \in \mathbb{R}^2 : y = x^2\}$ is a smooth manifold, so every supported valid inequality of $\mathcal{C}_1$ is exposed; $\mathcal{C}_2 := \{(x, y) \in \mathbb{R}^2 : y = |x|\}$ is smooth at $x \in [1, 2]$, so every supported valid inequality of $\mathcal{C}_2$ with supporting point (x, y) $(x \in [1, 2])$ is also exposed by the same point; however, a supported valid inequality of $\mathcal{C}_2$ with supporting point (x, y) $(x = 0)$ cannot be exposed, since there are infinitely many supported valid inequalities at the same point.*

The first theorem we present applies to full-dimensional nonconvex sets $\mathcal{S}$. We observed the geometric equivalence between the closed convex inner approximation of $\text{cl}(\mathcal{S}^c)$ and $\mathcal{S}$-free sets. The theorem provides a sufficient condition for the maximality of closed convex inner approximations.

Theorem 4.6. *Let $\mathcal{F}$ be a full-dimensional closed set in $\mathbb{R}^p$, and let $\mathcal{C} \subseteq \mathcal{F}$ be a full-dimensional closed convex set. If, for any $z^* \in \text{int}(\mathcal{F} \setminus \mathcal{C})$ and any $\lambda \in \text{dom}(\sigma_{\mathcal{C}})$ such that $\lambda \cdot z^* > \sigma_{\mathcal{C}}(\lambda)$, there exists a point $z' \in \text{bd}(\mathcal{F}) \cap \text{bd}(\mathcal{C})$ exposing $\lambda \cdot z \leq \sigma_{\mathcal{C}}(\lambda)$, then $\mathcal{C}$ is a maximal convex inner approximation of $\mathcal{F}$.*

Proof. Let $\mathcal{C}$ be a set satisfying the hypothesis. Suppose, to aim at a contradiction, that there exists a closed convex set $\mathcal{C}^*$ such that $\mathcal{C} \subsetneq \mathcal{C}^*$ and $\mathcal{C}^*$ is an inner approximation of $\mathcal{F}$. Then, there must exist an open ball B such that $B \subseteq \mathcal{F} \setminus \mathcal{C}$ and $B \subseteq \mathcal{C}^*$. Let z^* be the center of B, so $z^* \in \text{int}(\mathcal{F} \setminus \mathcal{C})$. W.l.o.g., we let $\mathcal{C}^* = \text{conv}(\mathcal{C} \cup \{z^*\})$, which is a closed convex

inner approximation of $\mathcal{F}$. Since $z^* \notin \mathcal{C}$, by the hyperplane separation theorem, there exists $\lambda \in \mathrm{dom}(\sigma_{\mathcal{C}})$ such that

$$\lambda \cdot z^* > \sigma_{\mathcal{C}}(\lambda). \tag{4.1}$$

For any such λ, by the hypothesis, there exists a point $z' \in \mathrm{bd}(\mathcal{F}) \cap \mathrm{bd}(\mathcal{C})$ such that

$$\lambda \cdot z' = \sigma_{\mathcal{C}}(\lambda), \tag{4.2}$$

and z' is an exposing point of $\mathcal{C}$. We want to show that, for any $\lambda' \in \mathrm{dom}(\sigma_{\mathcal{C}^*})$, $\lambda' \cdot z' < \sigma_{\mathcal{C}^*}(\lambda)$. We consider the following three cases. First, we consider the case $\lambda' = \lambda$. Because $z^* \in \mathcal{C}^*$, by the definition of support functions, we have that

$$\lambda \cdot z^* \leq \sup_{z \in \mathcal{C}^*} \lambda \cdot z = \sigma_{\mathcal{C}^*}(\lambda). \tag{4.3}$$

It follows from (4.1), (4.2), and (4.3) that

$$\lambda \cdot z' = \sigma_{\mathcal{C}}(\lambda) < \lambda \cdot z^* \leq \sigma_{\mathcal{C}^*}(\lambda) = \sigma_{\mathcal{C}^*}(\lambda'). \tag{4.4}$$

Second, we consider the case $\lambda' = \rho\lambda$ for some $\rho > 0$. Since $\sigma_{\mathcal{C}^*}$ is positively homogeneous of degree 1, it follows from (4.4) that $\lambda' \cdot z' = \rho\lambda \cdot z' < \rho\sigma_{\mathcal{C}^*}(\lambda) = \sigma_{\mathcal{C}^*}(\lambda')$. Last, we consider the case $\lambda' \in \mathrm{dom}(\sigma_{\mathcal{C}^*}) \setminus \{\rho\lambda\}_{\rho>0}$. By Lemma 4.3, $\sigma_{\mathcal{C}} \leq \sigma_{\mathcal{C}^*}$. By the hypothesis that z' is an exposing point of $\mathcal{C}$, provided that $\lambda' \neq \rho\lambda$, we have that $\lambda' \cdot z' < \sigma_{\mathcal{C}}(\lambda') \leq \sigma_{\mathcal{C}^*}(\lambda')$. In summary, we have proved that, for any $\lambda' \in \mathrm{dom}(\sigma_{\mathcal{C}^*})$, $\lambda' \cdot z' < \sigma_{\mathcal{C}^*}(\lambda')$. So by Lemma 4.3, $z' \in \mathrm{int}(\mathcal{C}^*)$. We find that $z' \in \mathrm{bd}(\mathcal{F}) \cap \mathrm{int}(\mathcal{C}^*)$. This finding means a point near z' exists, which is in $\mathcal{C}^*$, but not in $\mathcal{F}$. Hence, $\mathcal{C}^*$ is not an inner approximation of $\mathcal{F}$, which leads to a contradiction. □

We call z^* in Thm. 4.6 an *outlier point*, by which we try to enlarge an $\mathcal{S}$-free set, and let $L(z^*) := \{\lambda \in \mathrm{dom}(\sigma_{\mathcal{C}}) : \lambda \cdot z^* > \sigma_{\mathcal{C}}(\lambda)\}$. The proof of Thm. 4.6 was adapted from that of [233, Thm. 2], which excludes the presence of the outlier point and requires a stronger assumption, namely that, for any $\lambda \in \mathrm{dom}(\sigma_{\mathcal{C}})$ there exists a point $z' \in \mathrm{bd}(\mathcal{F}) \cap \mathrm{bd}(\mathcal{C})$ exposing $\lambda \cdot z \leq \sigma_{\mathcal{C}}(\lambda)$. As we will see in the proof of Thm. 4.8, $\bigcup_{z^* \in \mathrm{int}(\mathcal{F} \setminus \mathcal{C})} L(z^*)$ can be a proper subset of $\mathrm{dom}(\sigma_{\mathcal{C}})$, so we do not need to check that all $\lambda \in \mathrm{dom}(\sigma_{\mathcal{C}})$ are exposed.

We next focus on a specific type of function, namely *positive homogeneous functions*. We summarize their properties as follows.

Lemma 4.7. *Let f be a positive homogeneous function of degree $d \in \mathbb{R}$, such that for any $z \in \mathrm{dom}(f) \subseteq \mathbb{R}^p$ and any $\rho \in \mathbb{R}_{++}$, $f(\rho z) = \rho^d f(z)$. Then: (i) $\mathrm{int}(\mathrm{dom}(f))$ is a cone, and (ii) if $d = 1$, Then, for any $\breve{z} \in \mathrm{dom}(f)$, $\Xi^f_{\breve{z}}(z) = \nabla f(\breve{z}) \cdot z$ for $z \in \mathrm{dom}(f)$ and $\Xi^f_{\breve{z}}(z) = f(z)$ for $z = \rho\breve{z}$ with $\rho \in \mathbb{R}_{++}$.*

Proof. Given $z \in \mathrm{dom}(f)$, $f(\rho z) = \rho^d f(z)$ is a real number for any $\rho \in \mathbb{R}_{++}$, so $\mathrm{int}(\mathrm{dom}(f))$ is a cone. Suppose that f is positive homogeneous of degree 1. For any $z \in \mathrm{dom}(f)$, $\Xi^f_{\breve{z}}(z) = f(\breve{z}) + \nabla f(\breve{z}) \cdot (z - \breve{z}) = \nabla f(\breve{z}) \cdot z$, where the second equation follows from Euler's homogeneous function theorem: $f(\breve{z}) = \nabla f(\breve{z}) \cdot \breve{z}$. For any $z = \rho\breve{z}$ with $\rho \in \mathbb{R}_{++}$, $\Xi^f_{\breve{z}}(z) = \nabla f(\breve{z}) \cdot \rho\breve{z} = \rho\Xi^f_{\breve{z}}(\breve{z}) = \rho f(\breve{z}) = f(\rho\breve{z})$, where the first and second equations follow from the previous result, the third follows from that $\Xi^f_{\breve{z}}$ has the same value as f at $\breve{z}$, and the last equation follows from the homogeneity. □

We recall that $\Xi_{\breve{z}}^{f}$ in the above lemma is the first order linearization of f at $\breve{z}$. Moreover, $\mathrm{dom}(f)$ is embedded in $\mathbb{R}^p$, so we call $\mathbb{R}^p$ the *ambient space* of f.

The second theorem we present offers a more structured result specifically addressing nonconvex DCC sets $\mathcal{S}$. [272, Thm. 5.48] provides a sufficient condition for the maximality of the $\mathcal{S}$-free set described in Lemma 3.22. However, to clearly differentiate it from our subsequent result, we translate the condition into our setting as follows: (i) the functions f_1 and f_2 are superlinear, meaning they are positive homogeneous of degree 1 and super-additive (note that superlinear functions are concave), (ii) they are separable and act independently on different variables u and v, (iii) f_1 is negative everywhere except at 0, (iv) the linearization point $\tilde{v}$ of f_2 is nonzero, and (v) the domains $\mathrm{dom}(f_1)$ and $\mathrm{dom}(f_2)$ are Euclidean spaces.

Our second theorem provides an alternative condition for maximality that relaxes the condition (i) by requiring only one of f_1 or f_2 to be positive homogeneous of degree 1, while imposing mild regularity conditions. Additionally, it allows the domains to be full-dimensional convex cones.

Theorem 4.8. *For every $i \in \{1, 2\}$, let f_i be concave. Let $\mathcal{S} \coloneqq \{(u, v) \in \mathrm{dom}(f_1) \times \mathrm{dom}(f_2) : f_1(u) - f_2(v) \leq 0\}$. Suppose that: (i) at least one of f_1, f_2 is positive homogeneous of degree 1, (ii) f_1, f_2 are both positive/negative over the interiors of their domains, (iii) f_1 is continuously differentiable over $\mathrm{int}(\mathrm{dom}(f_1))$, and (iv) $\mathrm{dom}(f_1), \mathrm{dom}(f_2)$ are full-dimensional in the ambient spaces of f_1, f_2, respectively. Then, for any $\tilde{v} \in \mathrm{int}(\mathrm{dom}(f_2))$, $\mathcal{C} \coloneqq \{(u, v) \in \mathrm{dom}(f_1) \times \mathrm{dom}(f_2) : f_1(u) - \Xi_{\tilde{v}}^{f_2}(v) \geq 0\}$ is maximally $\mathcal{S}$-free in $\mathrm{dom}(f_1) \times \mathrm{dom}(f_2)$.*

Proof. We first adapt Lemma 3.22 by restricting the domain of z to the convex ground set $\mathcal{G} \coloneqq \mathrm{dom}(f_1) \times \mathrm{dom}(f_2)$. It follows from Lemma 3.22 that $\mathcal{C}$ is an $\mathcal{S}$-free set in $\mathcal{G}$. Since $\mathrm{dom}(f_1) \times \mathrm{dom}(f_2)$ are full-dimensional, $\mathcal{S}, \mathcal{C}, \mathcal{G}$ are full-dimensional. As $\mathcal{S}, \mathcal{C} \subseteq \mathcal{G}$, the maximality of $\mathcal{C}$ in $\mathcal{G}$ is equivalent to that $\mathcal{C}$ is a maximal convex inner approximation of $\mathcal{F} \coloneqq \mathrm{cl}(\mathcal{S}^c) \cap \mathcal{G} = \{(u, v) \in \mathcal{G} : f_1(u) - f_2(v) \geq 0\}$. Note that $\mathcal{F}$ is full-dimensional. We then apply Thm. 4.6 to prove that $\mathcal{C}$ is a maximal convex inner approximation of $\mathcal{F}$. Let $z^* \in \mathrm{int}(\mathcal{F} \setminus \mathcal{C})$ be any outlier point. It follows from the separating hyperplane theorem that there exists a supported valid inequality $\lambda \cdot z \leq \sigma_{\mathcal{C}}(\lambda)$ of $\mathcal{C}$ such that $\lambda \cdot z^* > \sigma_{\mathcal{C}}(\lambda)$. Since $\mathcal{F} \setminus \mathcal{C} \subseteq \mathcal{G}$, $\mathrm{int}(\mathcal{F} \setminus \mathcal{C}) \subseteq \mathcal{G}$. Since $\mathcal{C} \subseteq \mathcal{G}$, the inequality cannot be supported by a valid inequality at $\mathrm{bd}(\mathcal{G})$, so the inequality must be a valid inequality supported at $\mathcal{C} \setminus \mathrm{bd}(\mathcal{G})$. It follows from the concavity of f_1 that the inequality must admit the form $\Xi_{\breve{u}}^{f_1}(u) - \Xi_{\breve{v}}^{f_2}(v) \geq 0$ for some $\breve{u} \in \mathrm{dom}(f_1)$ (identical up to a positive multiplier). By the smoothness of f_1, w.l.o.g, we can perturb $\breve{u}$ such that it is in $\mathrm{int}(\mathrm{dom}(f_1))$. Let $\breve{v} \coloneqq \tilde{v}$. We now have that $\breve{u} \in \mathrm{int}(\mathrm{dom}(f_1)), \breve{v} \in \mathrm{int}(\mathrm{dom}(f_2))$. We will prove that $\Xi_{\breve{u}}^{f_1}(u) - \Xi_{\breve{v}}^{f_2}(v) \geq 0$ is exposed by a point $(u', v') \in (\mathrm{bd}(\mathcal{F}) \cap \mathrm{bd}(\mathcal{C})) \cap \mathrm{int}(\mathcal{G})$. It suffices to show that the following three equations hold:

$$\begin{aligned} \Xi_{\breve{u}}^{f_1}(u') - \Xi_{\breve{v}}^{f_2}(v') &= 0 \quad \text{(i.e., supported at } (u', v')\text{)}, \\ f_1(u') - \Xi_{\breve{v}}^{f_2}(v') &= 0 \quad \text{(i.e., } (u', v') \in \mathcal{C}\text{)}, \\ f_1(u') - f_2(v') &= 0 \quad \text{(i.e., } (u', v') \in \mathcal{F}\text{)}. \end{aligned} \tag{4.5}$$

Since $\mathcal{C} \subseteq \mathcal{F}$ and they are both full-dimensional, the last two equations imply that $(u', v') \in \mathrm{bd}(\mathcal{C}) \cap \mathrm{bd}(\mathcal{F})$. As f_1 is continuously differentiable and concave in the interior of its domain, the graph of $f_1(u) - \Xi_{\breve{v}}^{f_2}(v)$ over $\mathrm{int}(\mathcal{G})$ is a smooth manifold embedded in $\mathrm{int}(\mathcal{G}) \times \mathbb{R}$. The intersection of a smooth manifold with a hyperplane yields another lower-dimensional smooth manifold. This implies that the level set $\mathcal{C}$ of $f_1(u) - \Xi_{\breve{v}}^{f_2}(v)$ is also smooth at any point

$(u, v) \in \text{int}(\mathcal{G}) \cap \mathcal{C}$. By Obs. 4.5, (u, v) is an exposing point. Since $(u', v') \in \mathcal{C} \cap \text{int}(\mathcal{G})$, (u', v') is an exposing point, and the maximality of $\mathcal{C}$ is verified. We now proceed to construct (u', v') from $(\breve{u}, \breve{v})$ and prove (4.5). Let $\rho \coloneqq f_2(\breve{v})/f_1(\breve{u})$. Since $\breve{u} \in \text{int}(\text{dom}(f_1)), \breve{v} \in \text{int}(\text{dom}(f_2))$, by the assumption, $\rho > 0$. We consider the following two cases separately.

Case i. We first suppose that f_1 is positive homogeneous of degree 1. Let $(u', v') \coloneqq (\rho\breve{u}, \breve{v})$, which, by Lemma 4.7, is in $\text{int}(\mathcal{G})$. We have that:

$$f_1(u') \overset{(i.1)}{=} \Xi^{f_1}_{\breve{u}}(u') \overset{(i.2)}{=} \rho f_1(\breve{u}) \overset{(i.3)}{=} f_2(\breve{v}) \overset{(i.4)}{=} f_2(v') \overset{(i.5)}{=} \Xi^{f_2}_{\breve{v}}(v'),$$

where equations $(i.1), (i.2)$ follow from Lemma 4.7, $(i.3)$ follows from the definition of ρ, and $(i.4), (i.5)$ follow from $v' = \breve{v}$.

Case ii. We then suppose that f_2 is positive homogeneous of degree 1. Let $(u', v') \coloneqq (\breve{u}, \breve{v}/\rho) \in \text{int}(\mathcal{G})$. We have that:

$$\Xi^{f_1}_{\breve{u}}(u') \overset{(ii.1)}{=} f_1(u') \overset{(ii.2)}{=} f_1(\breve{u}) \overset{(ii.3)}{=} f_2(\breve{v})/\rho \overset{(ii.4)}{=} f_2(v') \overset{(ii.5)}{=} \Xi^{f_2}_{\breve{v}}(v'),$$

where equations $(ii.1), (ii.2)$ follow from $\breve{u} = u'$, $(ii.3)$ follows from the definition of ρ, and $(ii.4), (ii.5)$ follow from Lemma 4.7. Therefore, (4.5) are satisfied in both cases. □

We present the motivation for limiting the maximality of the set $\mathcal{C}$ within the ground set $\text{dom}(f_1) \times \text{dom}(f_2)$. The primary reason for this restriction stems from the difficulty in finding a non-trivial concave extension of f_1 over its ambient space such that for all $u \notin \text{dom}(f_1)$, $f_1(u) > -\infty$. While such an extension may exist geometrically, the construction of a closed-form expression remains unclear. In the next section, we will examine a specific example to illustrate this point.

Furthermore, we will employ the aforementioned theorem to develop DCC formulations of a nonconvex set. Notably, the functions f_1 and f_2 may not exhibit simultaneous positive homogeneity of degree 1, and their domains are non-negative orthants. Consequently, the relaxed condition on homogeneous degrees and domains in Thm. 4.8 becomes necessary. We give two examples to verify Thm. 4.8.

Example 4.9. *Let $f_1(u) \coloneqq u$ with $\text{dom}(f_1) \in \mathbb{R}$, and let $f_2(v) \coloneqq \sum_{i\in[n]} \sqrt{v_i}$ with $\text{dom}(f_2) = \mathbb{R}^n_+$. Note that f_1, f_2 are concave, $\text{dom}(f_2)$ is a non-negative orthant, and f_1 is positive homogeneous of degree 1. Let $\mathcal{G} \coloneqq \mathbb{R} \times \mathbb{R}^n_+$. One can verify that the presupposition of Thm. 4.8 is satisfied. Then, $\mathcal{S} \coloneqq \{(u, v) \in \mathcal{G} : u - \sum_{i\in[n]} \sqrt{v_i} \leq 0\}$ is a convex set. It is easy to see that $\mathcal{C} \coloneqq \{(u, v) \in \mathcal{G} : u - \sum_{i\in[n]} (\sqrt{\tilde{v}_i} + (v_i - \tilde{v}_i)/\sqrt{\tilde{v}_i}) \geq 0\}$ is maximally $\mathcal{S}$-free in $\mathcal{G}$ with $\tilde{v} > 0$.*

Example 4.10. *Exchange the functions f_1, f_2 in the previous examples. Then, $\mathcal{S} \coloneqq \{(u, v) \in \mathcal{G} : \sum_{i\in[n]} \sqrt{v_i} - u \leq 0\}$ is a reverse-convex set (i.e., the complement of a convex set). It is easy to see that $\mathcal{C} \coloneqq \{(u, v) \in \mathcal{G} : \sum_{i\in[n]} \sqrt{v_i} - u \geq 0\}$ is the unique maximal $\mathcal{S}$-free set in $\mathcal{G}$.*

4.1.3 $\mathcal{S}$-free sets in submodular optimization

We denote $\mathcal{B} \coloneqq \{0, 1\}^n$, $\bar{\mathcal{B}} \coloneqq [0, 1]^n$. We assume that $[n]$ is equipped with the natural number order. For $S \subseteq [n]$, we denote by $\text{supp}(S) \in \mathcal{B}$ the characteristic vector of S. Given a set $\mathcal{D} \subseteq \mathbb{R}^n$ and a function $g : \mathcal{D} \to \mathbb{R}$, we adopt the usual notation $\text{epi}_{\mathcal{D}}(g), \text{gr}_{\mathcal{D}}(g), \text{hypo}_{\mathcal{D}}(g)$ to denote the epigraph, graph and hypograph of g over $\mathcal{D}$, respectively. For example, $\text{gr}_{\mathcal{D}}(g) \coloneqq$

$\{(x,t) \in \mathcal{D} \times \mathbb{R} : g(x) = t\}$. When $\mathcal{D}$ is omitted in the subscript, it is assumed to be $\mathbb{R}^n$. We consider the submodular maximization problem:

$$\max_{t \in \mathbb{R}} t \quad \text{s.t.} \quad f(x) \geq t, \quad x \in \{0,1\}^n \cap \mathcal{X}. \tag{4.6}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a set describing additional constraints. The above formulation is the epigraphical reformulation of the submodular maximization problem $\max_{x \in \{0,1\}^n \cap \mathcal{X}} f(x)$. We study valid inequalities for the mixed-integer set $\mathrm{hypo}_{\{0,1\}^n}(f) := \{(x,t) \in \{0,1\}^n \times \mathbb{R} : f(x) \geq t\}$, which we call the hypograph of f over the Boolean hypercube $\{0,1\}^n$ (or, for brevity, the Boolean-hypograph of f).

The maximization of arbitrary submodular functions (i.e., Eq. (4.6)) can be reduced to a MILP with exponentially many linear inequalities [237]. The Benders-like exact approach based on a branch-and-cut algorithm proposed in [104] provides global dual bounds for primal solutions, and achieves a finite convergence rate.

Many submodular maximization problems (e.g., max cut with positive edge weights [270], D-optimal design [267], and utility maximization [8]) have natural MILP or MINLP formulations, which can be solved using general-purpose global optimization solvers. The algorithm underlying these solvers is typically a branch-and-cut algorithm, which uses polyhedral outer approximations to construct LP relaxations [61, 62, 284]. For submodular maximization problems with convex MINLP formulations, a state-of-art algorithm also uses polyhedral outer approximations [95].

The submodular maximization problem plays an intermediate role between these settings. On the one hand, the submodular function f is defined over the Boolean hypercube $\{0,1\}^n$. Therefore, the graph of f projected on $\mathbb{R}^n$ is a subset of a lattice. On the other hand, as a discrete analogue to convex functions, f has a convex (thus continuous) extension over the hypercube $[0,1]^n$, namely the Lovász extension [212]. We can extend the Lovász extension to a convex function, which we call $\bar{\mathsf{F}}_f$, over the entire n-dimensional Euclidean space $\mathbb{R}^n$. This (continuous) function $\bar{\mathsf{F}}_f$ inherits a rich combinatorial structure from f.

The difference of two submodular functions (call them f_1, f_2) is a *submodular-supermodular* (SS) function. SS functions generalize submodular functions, which are also discrete analogs of difference-of-convex (DC) functions. In fact, SS functions may represent some discrete nonconvex functions arising in combinatorial optimization. For example, we will show that any Boolean multilinear function is an SS function.

We denote $\mathcal{B} := \{0,1\}^n$, $\bar{\mathcal{B}} := [0,1]^n$. We assume that $[n]$ is equipped with the natural number order. For $S \subseteq [n]$, we denote by $\mathsf{supp}(S) \in \mathcal{B}$ the characteristic vector of S. Given a set $\mathcal{D} \subseteq \mathbb{R}^n$ and a function $g : \mathcal{D} \to \mathbb{R}$, we adopt the usual notation $\mathrm{epi}_{\mathcal{D}}(g), \mathrm{gr}_{\mathcal{D}}(g), \mathrm{hypo}_{\mathcal{D}}(g)$ to denote the epigraph, graph and hypograph of g over $\mathcal{D}$, respectively. For example, $\mathrm{gr}_{\mathcal{D}}(g) := \{(x,t) \in \mathcal{D} \times \mathbb{R} : g(x) = t\}$. When $\mathcal{D}$ is omitted in the subscript, it is assumed to be $\mathbb{R}^n$.

Extensions of submodular functions

We study continuous extensions of submodular functions. W.l.o.g., we assume in the sequel that, for any submodular function f, $f(\mathbf{0}) = 0$ holds (by a translation of a constant). It is known that the Lovász extension [212] extends f from $\mathcal{B}$ to $\bar{\mathcal{B}}$. Based on this extension, we construct another extension $\bar{\mathsf{F}}_f$ of f defined over the entire space $\mathbb{R}^n$, and study its analytical and combinatorial structures.

We first look at some polyhedra associated with the submodular function f [23, 270]. Its *extended polymatroid* is defined as

$$\mathsf{EPM}_f := \{s \in \mathbb{R}^n : \forall x \in \mathcal{B},\, sx \leq f(x)\}, \tag{4.7}$$

and the convex hull of the Boolean-epigraph f over $\mathcal{B}$ is defined as

$$Q_f := \mathrm{conv}(\mathrm{epi}_{\mathcal{B}}(f)).$$

Recall that $\mathrm{ext}(\mathsf{EPM}_f)$ are the vertices of EPM_f. We further define the polyhedron

$$\mathsf{EE}_f := \{(x,t) \in \mathbb{R}^{n+1} : \forall s \in \mathrm{ext}(\mathsf{EPM}_f),\, sx \leq t\}. \tag{4.8}$$

In fact, EE_f contains Q_f, because of the following lemma:

Lemma 4.11 ([23])**.** $Q_f = \mathsf{EE}_f \cap (\bar{\mathcal{B}} \times \mathbb{R})$.

Therefore, $x \in \bar{\mathcal{B}}$ defines trivial facets of Q_f, and non-trivial facets of Q_f are $sx \leq t$, where s is a vertex of EPM_f.

These polyhedra in turn give rise to some functions associated with f. Since Q_f is the epigraph of F_f, by Lemma 4.11,

$$\mathsf{F}_f : \bar{\mathcal{B}} \to \mathbb{R}, \quad x \mapsto \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} sx. \tag{4.9}$$

We remark that F_f is equivalent to the Lovász extension of f [23]. We will show that the cardinality $|\mathrm{ext}(\mathsf{EPM}_f)|$ is not polynomial in n. Thus, when computing F_f, it is inefficient to evaluate all sx for $s \in \mathrm{ext}(\mathsf{EPM}_f)$. However, the value and the (sub)-gradients of F_f at points in $\bar{\mathcal{B}}$ can be computed in a strongly polynomial time (see Sect. 3.2.1).

We define the envelope of f extended to $\mathbb{R}^n$ as

$$\bar{\mathsf{F}}_f : \mathbb{R}^n \to \mathbb{R}, \quad x \mapsto \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} sx. \tag{4.10}$$

We note that $\bar{\mathsf{F}}_f$ simply enlarges the domain of F_f from $\bar{\mathcal{B}}$ to $\mathbb{R}^n$. This extension is algebraically simple, but analytically less so. The analytical properties of $\bar{\mathsf{F}}_f(x)$ outside $\bar{\mathcal{B}}$ will be studied in further detail. We find that EE_f is the epigraph of $\bar{\mathsf{F}}_f$, i.e., $\mathsf{EE}_f = \mathrm{epi}(\bar{\mathsf{F}}_f)$, so $\bar{\mathsf{F}}_f$ is a convex function. Since every facet $sx \leq t$ of EE_f is in one-to-one correspondence to a linear underestimator function sx of $\bar{\mathsf{F}}_f$, we call EE_f the *extended envelope epigraph*.

The problem of efficiently evaluating $\bar{\mathsf{F}}_f$ and its associated sub-gradients at a point in $\mathbb{R}^n$ is very important, because it is crucial in constructing intersection cuts. Regarding $\bar{\mathsf{F}}_f$, one can compute its value and sub-gradients at points in $\bar{\mathcal{B}}$ in a strongly polynomial time using a sorting algorithm. As the Lovász extension F_f is a restriction of $\bar{\mathsf{F}}_f$ to the hypercube $\bar{\mathcal{B}}$, this fact implies that many properties of $\bar{\mathsf{F}}_f$ may also hold for $\bar{\mathsf{F}}_f$. In the following, we will show how we can reuse the sorting algorithm to compute $\bar{\mathsf{F}}_f$ over the entire space $\mathbb{R}^n$. This extension requires us to study the properties of $\bar{\mathsf{F}}_f$ and EE_f.

By Lemma 4.11, one may observe that a facet of EE_f includes and extends geometrically a facet of Q_f. This observation reveals the close relation between F_f in (4.9) and $\bar{\mathsf{F}}_f$ in (4.10). In order to separate facets of EE_f, we consider $\bar{\mathsf{F}}_f$ since EE_f is its epigraph.

Given $\tilde{x} \in \mathbb{R}^n$, the evaluation of $\bar{\mathsf{F}}_f(\tilde{x})$ is called the *extended polymatroid vertex maximization problem*, as by definition $\bar{\mathsf{F}}_f(\tilde{x})$ equals

$$\max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s\tilde{x}. \tag{4.11}$$

Any optimal solution s^* of Eq. (4.11) is a subgradient of $\bar{\mathsf{F}}_f$ at $\tilde{x}$, i.e., $s^* \in \partial \bar{\mathsf{F}}_f(\tilde{x})$. The set of vertices $\mathrm{ext}(\mathsf{EPM}_f)$ is the image of S_n under the map σ.

Lemma 4.12 ([133])**.** $\sigma(S_n) = \mathrm{ext}(\mathsf{EPM}_f)$.

By Lemma 4.12, $\max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s\tilde{x} = \max_{\pi \in S_n} \sigma(\pi)\tilde{x}$, so (4.11) asks for a permutation π^* that maximizes $\sigma(\pi^*)\tilde{x}$.

To tackle (4.11), we look at a related relaxed problem, namely the *extended polymatroid maximization problem*:

$$\max_{s \in \mathsf{EPM}_f} s\tilde{x}, \tag{4.12}$$

which is equivalent to the problem (3.23) due to Lemma 4.12.

We note that the vertices $\mathrm{ext}(\mathsf{EPM}_f)$ are a finite set, so (4.11) is always bounded. Moreover, by the Minkowski-Weyl theorem [100], EPM_f is the Minkowski sum of the polytope $\mathrm{conv}(\mathrm{ext}(\mathsf{EPM}_f))$ and the recession cone of EPM_f. By Proposition 3.15 of [100] and (4.7), the recession cone admits the form $\{s \in \mathbb{R}^n : \forall x \in \mathcal{B},\, sx \leq 0\}$, so the cone is non-empty, and EPM_f is unbounded. This means that (4.12) may be unbounded.

Lemma 4.13 ([23, 133])**.** *When $\tilde{x} \geq 0$, the optimum of* (4.12) *is a vertex of* EPM_f *(i.e., in* $\mathrm{ext}(\mathsf{EPM}_f)$*), and* (4.11) *is equivalent to* (4.12)*; when $\tilde{x}$ has some negative entries,* (4.11) *is unbounded, and therefore not equivalent to* (4.12)*.*

Even if (4.11) is not equivalent to (4.12) in general, we show that (4.11) can still be solved by the sorting algorithm.

Proposition 4.14. *The output of the sorting algorithm is an optimal solution of the extended polymatroid vertex maximization problem* (4.11)*.*

Proof. Let π^* be the permutation found by the sorting algorithm. By Lemma 4.12, $\sigma(\pi^*)$ is in $\mathrm{ext}(\mathsf{EPM}_f)$ and hence a feasible solution to (4.11). Next, we prove the optimality of $\sigma(\pi^*)$. Let the scalar $d := \min_{i \in [n]} \tilde{x}_i$. We can write $\tilde{x}$ as the sum of $(\tilde{x} - d\mathbf{1}), d\mathbf{1}$, where the translated vector $\tilde{x} - d\mathbf{1} = (\tilde{x}_i - d)_{i \in [n]}$ has non-negative components. We find that the following inequalities hold:

$$\begin{aligned} \sigma(\pi^*)\tilde{x} &\leq \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s\tilde{x} = \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s(\tilde{x} - d\mathbf{1} + d\mathbf{1}) \\ &\leq \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s(\tilde{x} - d\mathbf{1}) + \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s(d\mathbf{1}), \end{aligned} \tag{4.13}$$

where the first inequality follows from the fact given by Lemma 4.12 that $\sigma(\pi^*)$ is in $\mathrm{ext}(\mathsf{EPM}_f)$, the last inequality follows from the fact that maximum of the sum is at most the sum of maxima. We next construct the optimal solutions to $\max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s(\tilde{x} - d\mathbf{1})$ and $\max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s(d\mathbf{1})$, respectively. First, since the permutation π^* maps $\tilde{x}$ into a vector with non-increasing entries and the entries of $d\mathbf{1}$ are identical, we have that $(\tilde{x} - d\mathbf{1})_{\pi^*(1)} \geq \cdots \geq (\tilde{x} - d\mathbf{1})_{\pi^*(n)}$. Since $\tilde{x} - d\mathbf{1}$ is constructed non-negative, by Lemma 4.13, $\max_{s \in \mathsf{EPM}_f} s(\tilde{x} - d\mathbf{1}) = \max_{s \in \mathrm{ext}(\mathsf{EPM}_f)} s(\tilde{x} - d\mathbf{1})$. Moreover, π^* is also the permutation that sorts $(\tilde{x} - d\mathbf{1})$ in a non-increasing order, it follows

again from Lemma 4.13 that $\sigma(\pi^*)$ is an optimal solution to $\max_{s\in\mathsf{EPM}_f} s(\tilde{x}-d\mathbf{1})$. This implies that $\sigma(\pi^*)$ is also an optimal solution to $\max_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s(\tilde{x}-d\mathbf{1})$. Secondly, for any $\pi\in S_n$, it follows from Defn. 3.12 that $v^n(\pi)=\mathbf{1}$. This implies that $\sigma(\pi)v^n(\pi)=\sigma(\pi)\mathbf{1}=f(\mathbf{1})$, where the last equation follows from Cor. 3.14. In addition, by Lemma 4.12, $\mathrm{argmax}_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s(d\mathbf{1}) = \sigma(\mathrm{argmax}_{\pi\in S_n}\sigma(\pi)(d\mathbf{1}))$. Since all $\sigma(\pi)(d\mathbf{1})$ are identically equal to $df(\mathbf{1})$, we can pick $\sigma(\pi^*)$ as the optimal solution to $\max_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s(d\mathbf{1})$. Finally, we find that $\max_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s(\tilde{x}-d\mathbf{1})$ and $\max_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s(d\mathbf{1})$ have a common optimal solution $\sigma(\pi^*)$. This implies that the inequalities in (4.13) become equations, because

$$\sigma(\pi^*)\tilde{x}\le\max_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s\tilde{x}\le\sigma(\pi^*)(\tilde{x}-d\mathbf{1})+\sigma(\pi^*)(d\mathbf{1})=\sigma(\pi^*)\tilde{x}.$$

Therefore, $\sigma(\pi^*)$ is an optimal solution to $\max_{s\in\mathrm{ext}(\mathsf{EPM}_f)} s\tilde{x}$.

□

Given $\tilde{x}\in\mathbb{R}^n$, the sorting algorithm outputs a permutation acting on the entries of . The sorting algorithm is translation-invariant, i.e., translating each entry of $\tilde{x}$ by the same value does not change the output permutation. A by-product of Prop. 4.14 is that $\bar{\mathsf{F}}_f$ is linear over specific lines specified as follows.

Corollary 4.15. *Let $\tilde{x}\in\mathbb{R}^n$, then $\bar{\mathsf{F}}_f$ is linear on $\tilde{x}+\lambda\mathbf{1}$ w.r.t. $\lambda\in\mathbb{R}$.*

We look at the boundary of EE_f. By Cor. 3.14 and Cor. 3.15, for all $x\in\mathcal{B}$, the point $(x,f(x))$ supports some facets of EE_f.

Theorem 4.16. $\mathsf{EE}_f\cap\mathrm{hypo}_{\mathcal{B}}(f)=\mathrm{gr}_{\mathcal{B}}(f)\subseteq\mathrm{bd}(\mathsf{EE}_f)$.

Proof. We consider a point $v\in\mathcal{B}$ and look at the line $\ell=\{(v,t):t\in\mathbb{R}\}$. It can be separated into the restricted epigraph $\ell_+:=\{(v,t):f(v)\le t\}$ and the restricted hypograph $\ell_-:=\{(v,t):f(v)\ge t\}$, as $\ell_+\cap\ell_-=(v,f(v))$ and $\ell=\ell_+\cup\ell_-$. First, we know that, by definition of Q_f and Lemma 4.11, $\ell_+\subseteq Q_f\subseteq\mathsf{EE}_f$. Second, by Cor. 3.14, the point $(v,f(v))$ supports some facets of EE_f, so the point (v,t) with $t<f(v)$ is separated by these facets from EE_f. Thereby, we know that $\ell_-\cap\mathsf{EE}_f=\{(v,f(v))\}$. To summarize, we know that $\mathsf{EE}_f\cap\ell=\ell_+$ and $(v,f(v))\in\mathrm{bd}(\mathsf{EE}_f)$. As $\mathrm{gr}_{\mathcal{B}}(f)=\bigcup_{v\in\mathcal{B}}\{(v,f(v))\}$, we have that $\mathrm{gr}_{\mathcal{B}}(f)\subseteq\mathrm{bd}(\mathsf{EE}_f)$. As the hypograph $\mathrm{hypo}_{\mathcal{B}}(f)=\bigcup_{v\in\mathcal{B}}\{(v,t):f(v)\ge t\}$ (union of restricted hypographs), we have that $\mathsf{EE}_f\cap\mathrm{hypo}_{\mathcal{B}}(f)=\mathrm{gr}_{\mathcal{B}}(f)$. □ □

As mentioned above, $\bar{\mathsf{F}}_f$ is convex and $\mathsf{EE}_f=\mathrm{epi}(\bar{\mathsf{F}}_f)$, so $\bar{\mathsf{F}}_f$ is also a continuous extension of f. As EE_f contains Q_f, $\bar{\mathsf{F}}_f$ further extends F_f (the Lovász extension).

We now understand enough of the facial structure of EE_f that we can construct Boolean-hypograph-free sets. We can also compute the value and subgradients of $\bar{\mathsf{F}}_f$ at any point in $\mathbb{R}^n$, which are used in the construction of intersection cuts.

Boolean-hypograph-free sets for submodular functions

We consider two types of Boolean-hypograph-free sets for a given submodular function f.

First, we show that one can lift a maximal $\mathcal{B}$-free set into a maximal $\mathrm{hypo}_{\mathcal{B}}(f)$-free set.

Theorem 4.17. *Let $f:\mathcal{B}\to\mathbb{R}$ be an arbitrary function, and let $\mathcal{K}$ be a maximal $\mathcal{B}$-free set in $\mathbb{R}^n$. Then $\mathcal{C}:=\mathcal{K}\times\mathbb{R}$ is a maximal $\mathrm{hypo}_{\mathcal{B}}(f)$-free set.*

Proof. We note that $\text{int}(\mathcal{C}) = \text{int}(\mathcal{K}) \times \mathbb{R}$. Since $\text{int}(\mathcal{C}) \cap \text{hypo}_{\mathcal{B}}(f) = \varnothing$, $\mathcal{C}$ is $\text{hypo}_{\mathcal{B}}(f)$-free. Assume that there exists a $\text{hypo}_{\mathcal{B}}(f)$-free set $\mathcal{C}'$ containing $\mathcal{C}$. Then the recession cone of $\mathcal{C}'$ must contain that of $\mathcal{C}$, so $\mathcal{C}' = \mathcal{K}' \times \mathbb{R}$ for some closed convex set $\mathcal{K}'$ containing $\mathcal{K}$. Moreover, $\mathcal{K}'$ must be a $\mathcal{B}$-free set, otherwise, there exists a point $x \in \mathcal{B} \cap \text{int}(\mathcal{K}')$ such that $(x, f(x)) \in \text{int}(\mathcal{K}') \times \mathbb{R} = \text{int}(\mathcal{C}')$. However, since $\mathcal{K}$ is maximally $\mathcal{B}$-free, this implies that $\mathcal{K} = \mathcal{K}'$. As a result, $\mathcal{C} = \mathcal{C}'$, so $\mathcal{C}$ is maximal. □

This construction does not rely on any structure of f, as it just lifts a $\mathcal{B}$-free set. For any $j \in [n]$, the simple lifted split $\{x \in \mathbb{R}^n : 0 \leq x_j \leq 1\} \times \mathbb{R}$ is a maximal $\text{hypo}_{\mathcal{B}}(f)$-free set. We next construct $\text{hypo}_{\mathcal{B}}(f)$-free sets using submodularity, for both theoretical and computational interests. We show that both the extended epigraph EE_f and its strict subset Q_f are Boolean-hypograph-free sets.

Proposition 4.18. EE_f, Q_f *are* $\text{hypo}_{\mathcal{B}}(f)$*-free sets.*

Proof. Since $\text{gr}_{\mathcal{B}}(f) \subseteq \text{bd}(\text{EE}_f)$, we conclude that $\text{EE}_f \cap \text{hypo}_{\mathcal{B}}(f) \subseteq \text{bd}(\text{EE}_f)$ and hence $\text{int}(\text{EE}_f) \cap \text{hypo}_{\mathcal{B}}(f) = \varnothing$. Additionally, EE_f is convex and hence $\text{hypo}_{\mathcal{B}}(f)$-free. As $Q_f \subseteq \text{EE}_f$, Q_f is $\text{hypo}_{\mathcal{B}}(f)$-free set. □

It is known that the maximal Boolean-hypograph-free set of a convex function is its epigraph. We shall show, however, that the extended epigraph EE_f of a submodular function f is not a maximal Boolean-hypograph-free set. At a high-level, a possible way to test the maximality of EE_f is as follows. The set Q_f is the convex hull of $\text{epi}_{\mathcal{B}}(f)$. Geometrically, Q_f is the "minimal" convex set containing $\text{epi}_{\mathcal{B}}(f)$. Intuitively, it is unlikely that a "minimal" set turns out to be a good "maximal" $\text{hypo}_{\mathcal{B}}(f)$-free set. We therefore remove some facets from Q_f in order to enlarge this polyhedron. After removing trivial facets of Q_f, the enlarged polyhedron is the extended epigraph EE_f of the envelope of f. However, this enlargement is still not sufficient. We therefore look at a further enlargement of EE_f.

The following fundamental theorem gives a sufficient and necessary condition on (maximal) Boolean-hypograph-free sets containing EE_f.

Theorem 4.19. *Let $\mathcal{C}$ be a full-dimensional closed convex set in $\mathbb{R}^{n+1}$ containing EE_f. Then $\mathcal{C}$ is a $\text{hypo}_{\mathcal{B}}(f)$-free set if and only if $\mathcal{C}$ is $\text{gr}_{\mathcal{B}}(f)$-free. Moreover, $\mathcal{C}$ is a maximal $\text{hypo}_{\mathcal{B}}(f)$-free set if and only if $\mathcal{C}$ is a polyhedron and there is at least one point of $\text{gr}_{\mathcal{B}}(f)$ in the relative interior of each facet of $\mathcal{C}$.*

Proof. We note that by Thm. 4.16, $\text{gr}_{\mathcal{B}}(f) \subseteq \text{bd}(\text{EE}_f) \subseteq \text{EE}_f \subseteq \mathcal{C}$. Thereby, $\text{gr}_{\mathcal{B}}(f) \cap \text{int}(\mathcal{C}) = \varnothing$ (i.e., $\mathcal{C}$ is $\text{gr}_{\mathcal{B}}(f)$-free) if and only if $\text{gr}_{\mathcal{B}}(f) \subseteq \text{bd}(\mathcal{C})$.

We consider the $\mathcal{S}$-freeness first. We prove the forward direction. Assume that $\mathcal{C}$ is a $\text{hypo}_{\mathcal{B}}(f)$-free set. Suppose, to aim at a contradiction, that there exists a point $(v, f(v)) \in \text{int}(\mathcal{C}) \cap \text{gr}_{\mathcal{B}}(f)$. Then there exists a sufficiently small $\epsilon > 0$ such that $(v, f(v) - \epsilon) \in \text{int}(\mathcal{C})$, but $(v, f(v) - \epsilon) \in \text{hypo}_{\mathcal{B}}(f)$, which leads to a contradiction. We prove the reverse direction. Assume that $\mathcal{C}$ is $\text{gr}_{\mathcal{B}}(f)$-free. Suppose, to aim at a contradiction, that there exists a point $(v, f(v)-\delta) \in \text{int}(\mathcal{C})$ with $v \in \mathcal{B}$ and $\delta > 0$. As, for some $\epsilon > 0$, $(v, f(v)+\epsilon) \subseteq \text{int}(\text{EE}_f) \subseteq \text{int}(\mathcal{C})$, by convexity of $\mathcal{C}$, $(v, f(v)) \in \text{int}(\mathcal{C})$, which leads to a contradiction. This implies that $\mathcal{C}$ is $\text{hypo}_{\mathcal{B}}(f)$-free if and only if $\text{gr}_{\mathcal{B}}(f)$-free (or $\text{gr}_{\mathcal{B}}(f) \subseteq \text{bd}(\mathcal{C})$).

We consider maximality next. The proof is similar to that [38] for the bounded maximal lattice-free set. Let $\mathcal{C}$ be a maximal $\text{gr}_{\mathcal{B}}(f)$-free set. For each $v \in \text{gr}_{\mathcal{B}}(f)$, it follows from the separating hyperplane theorem that there exists a half-space $\{z : a^v z \leq b_v\}$ containing $\mathcal{C}$

such that $a^v v = b_v$. As $\mathrm{gr}_{\mathcal{B}}(f)$ is a finite set, the set $P := \{z : \forall v \in \mathrm{gr}_{\mathcal{B}}(f), a^v z \leq b_v\}$ is a polyhedron. By construction, P is $\mathrm{gr}_{\mathcal{B}}(f)$-free and $\mathcal{C} \subseteq P$, thus $\mathcal{C} = P$ by maximality of $\mathcal{C}$.

We now show that there is at least one point of $\mathrm{gr}_{\mathcal{B}}(f)$ in the relative interior of each facet of $\mathcal{C}$. Assume $\mathcal{C} = \{z : \forall i \in M, a^i z \leq b_i\}$, where $a^i z \leq b_i, i \in M$, are all distinct facet-defining inequalities for $\mathcal{C}$. Suppose, to aim at a contradiction, that the facet $F_t = \{z \in \mathcal{C} : a^t z = b_t\}$ does not contain any point of $\mathrm{gr}_{\mathcal{B}}(f)$ in its relative interior. Let $\epsilon > 0$, enlarge $\mathcal{C}$ to another polyhedron $\mathcal{C}' := \{z : \forall i \in M \setminus \{t\}, a^i z \leq b_i, a^t z \leq b_t + \epsilon\}$. As $\mathcal{C} \subsetneq \mathcal{C}'$ and $\mathcal{C}$ is maximally $\mathrm{gr}_{\mathcal{B}}(f)$-free, $\mathcal{C}'$ contains points of $\mathrm{gr}_{\mathcal{B}}(f)$ in its interior. Thus, the point $z' := \mathrm{argmin}_{z \in \mathrm{int}(\mathcal{C}') \cap \mathrm{gr}_{\mathcal{B}}(f)} a^t z$ exists. It follows from $z' \in \mathrm{int}(\mathcal{C}')$ that $\forall i \in M \setminus \{t\}$, $a^i z' < b_i$, and $a^t z' < b_t + \epsilon$. Since $F_t = \{z : \forall i \in M \setminus \{t\}, a^i z \leq b_i, a^t z = b_t\}$, $a^t z'$ cannot equal b_t, otherwise, this implies that F_t contains $z' \in \mathrm{gr}_{\mathcal{B}}(f)$ in its relative interior. Since $\mathcal{C}$ is $\mathrm{gr}_{\mathcal{B}}(f)$-free, $a^t z'$ cannot be strictly less than b_t, otherwise, this implies that $\mathcal{C}$ contains $z' \in \mathrm{gr}_{\mathcal{B}}(f)$. Then, it must be that $a^t z' > b_t$, and $\mathcal{C}^* := \{\forall i \in M \setminus \{t\}, a^i z \leq b_i, a^t z \leq a^t z'\}$ strictly includes $\mathcal{C}$. By construction, $\mathcal{C}^*$ does not contain any point of $\mathrm{gr}_{\mathcal{B}}(f)$ in its interior. This contradicts the maximality of $\mathcal{C}$. □ □

The above theorem is purely geometrical. Since submodular functions are combinatorial objects, we translate this theorem to a combinatorial language. We first define a combinatorial object in the Boolean hypercube $\mathcal{B}$.

Definition 4.20. *Let $x^0, x^1, \ldots, x^n$ be $n+1$ distinct points of $\mathcal{B}$. They are called monotone, if $\mathbf{0} = x^0 < x^1 < \cdots < x^n = \mathbf{1}$. We call the corresponding ordered set $(x^0, \ldots, x^n) \subseteq \mathcal{B}$ a monotone chain in $\mathcal{B}$.*

Therefore, we use a monotone chain to represent a set of monotone points. Then we have the following observation.

Proposition 4.21. *The set S_n of permutations is in one-to-one correspondence to the set of monotone chains via the map V defined as follows: for all $\pi \in S_n$, $V(\pi) := (v^i(\pi) \mid i \in \mathcal{N} \cup \{0\})$.*

Proof. It suffices to prove that, under the map, each permutation is mapped to a monotone chain, and for each monotone chain, there exists a permutation mapped to it. By Cor. 3.14, since $\varnothing = \pi([0]) \subsetneq \cdots \subsetneq \pi([n]) = [n]$, by Defn. 3.12, $0 = v^0(\pi) < \cdots < v^n(\pi) = 1$, so $V(\pi)$ is a monotone chain. Conversely, given a monotone chain $(x^0, \ldots, x^n)$, we construct π as follows: $\pi(0) = 0$; and for all $i \in [n]$, $\pi(i)$ is the index of the unique non-zero entry of $x^i - x^{i-1}$. It follows that $V(\pi)$ is the chain. □

We find that permutations and monotone chains are indeed equivalent. We note that any $n+1$ distinct points from $\mathrm{gr}_{\mathcal{B}}(f)$ are affinely independent in $\mathbb{R}^{n+1}$ and hence support a hyperplane in $\mathbb{R}^{n+1}$. Thereby, we can infer from Cor. 3.14 and Prop. 4.21 that

Corollary 4.22. *If $(x^0, \ldots, x^n)$ is a monotone chain in $\mathcal{B}$, then distinct points $(x^0, f(x^0)), \ldots, (x^n, f(x^n))$ of $\mathrm{gr}_{\mathcal{B}}(f)$ define (or support) a facet of the extended envelope epigraph EE_f.*

We say that this monotone chain *induces the facet*. In fact, we find that facets of EE_f, permutations on $[n]$, and monotone chains in $\mathcal{B}$ are in one-to-one correspondence. Therefore, we can view them as the same objects. In particular, Prop. 4.21 relates permutations and monotone chains. We give the following characterization of permutations on $[n]$.

Definition 4.23. *A subset S'_n of permutations of S_n is called a cover, if $\bigcup_{\pi \in S'_n} V(\pi) = \mathcal{B}$; moreover, S'_n is called a minimal cover if, additionally, for all $\pi \in S'_n$, $V(\pi) \setminus \bigcup_{\pi' \in S'_n : \pi' \neq \pi} V(\pi')$ is not empty.*

We want to enlarge EE_f by removing its facets. This is equivalent to removing permutations from S_n. Let S'_n be a subset of permutations of S_n, and $\mathcal{C}(S'_n) := \{(x,t) : \forall \pi \in S'_n,\, \sigma(\pi)x \leq t\}$ denotes the relaxation of the extended envelope epigraph induced by S'_n. It is obvious that $\mathsf{EE}_f = \mathcal{C}(S_n) \subseteq \mathcal{C}(S'_n)$ for any $S'_n \subseteq S_n$. The following corollary translates Thm. 4.19 to a combinatorial language.

Corollary 4.24. *Let S'_n be a subset of permutations of S_n. $\mathcal{C}(S'_n)$ is $\mathrm{hypo}_{\mathcal{B}}(f)$-free if and only if S'_n is a cover. $\mathcal{C}(S'_n)$ is maximally $\mathrm{hypo}_{\mathcal{B}}(f)$-free if and only if S'_n is a minimal cover.*

Proof. First, we note that $C(S'_n)$, as a relaxation of EE_f contains $\mathrm{gr}_{\mathcal{B}}(f)$. Next, we assume that S'_n is a cover. Then points of $\mathrm{gr}_{\mathcal{B}}(f)$ support facets of $C(S'_n)$. By Thm. 4.19, $C(S'_n)$ is $\mathrm{hypo}_{\mathcal{B}}(f)$-free if and only if it is a cover. Finally, S'_n is a minimal cover, if and only if then each facet of $C(S'_n)$ has a point of $\mathrm{gr}_{\mathcal{B}}(f)$ in its interior. By Thm. 4.19, the later is equivalent to that $C(S'_n)$ is maximally $\mathrm{hypo}_{\mathcal{B}}(f)$-free. □

We can now disprove the maximality EE_f by means of a counter-example. Thanks to Cor. 4.24, we can use a counting argument to show that we can remove facets from EE_f. This results in a new enlarged $\mathrm{hypo}_{\mathcal{B}}(f)$-free polyhedron.

Proposition 4.25. EE_f *is not maximally Boolean-hypograph-free.*

Proof. It suffices to find a counter-example. Consider $n = 3$, $\mathcal{B} = \{0,1\}^3$, there are 6 permutations, and 6 monotone chains (see Fig. 4.1). We assume that, in a non-degenerate case, the associated extended envelope epigraph EE_f has 6 facets induced by 6 chains respectively. The vertices $(0,0,0)$ and $(1,1,1)$ are visited by all the chains, while the other vertices are visited twice each. Therefore, a chain cannot "exclusively" visit a vertex, so the corresponding facet cannot contain one point of $\mathrm{gr}_{\mathcal{B}}(f)$ in its relative interior. In fact, we can remove some facets from the extended envelope epigraph. We keep three chains:

$$
\begin{aligned}
&(\,(0,0,0),(0,0,1),(0,1,1),(1,1,1)\,)\,,\\
&(\,(0,0,0),(0,1,0),(1,1,0),(1,1,1)\,)\,,\\
&(\,(0,0,0),(1,0,0),(1,0,1),(1,1,1)\,)\,.
\end{aligned}
$$

These chains induce 3 facets such that at least one point of $\mathrm{gr}_{\mathcal{B}}(f)$ is in the relative interior of each facet and each point of $\mathcal{B}$ is in these 3 facets, so the polyhedron defined by these 3 facets is a $\mathrm{hypo}_{\mathcal{B}}(f)$-free set larger than EE_f. □

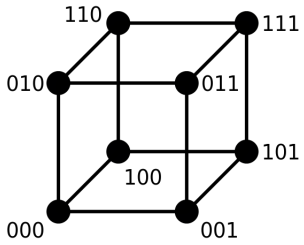


Figure 4.1 a Boolean cube $\mathcal{B} = \{0,1\}^3$ © [134]

We discuss why enlarging EE_f is not a trivial feat. We build a bipartite graph $G := (\mathcal{B} \cup S_n, E)$. An edge e of E connects a vertex $v \in \mathcal{B}$ to a permutation $\pi \in S_n$ if $v \in V(\pi)$.

Then, a minimal cover is a subset S'_n of S_n such that: i) each vertex of $\mathcal{B}$ is incident to at least one permutation in S'_n; ii) each permutation in S'_n is incident to a vertex of $\mathcal{B}$ that no other permutation in S'_n is incident to. As $|\mathcal{B}| = 2^n$ and $|S_n| = n!$, the size of such a graph is not polynomial in n. We need additional information in order to enlarge EE_f efficiently.

We relax the submodular maximization problem (4.6) through a polyhedral outer approximation $\mathcal{P}$ of $\mathrm{hypo}_{\mathcal{B}}(f)$. Let X be the orthogonal projection of $\mathcal{P}$ on x-space. We remark that, within a branch-and-cut algorithm, X might be within a low-dimensional face of $\bar{\mathcal{B}}$. Let $\tilde{z} := (\tilde{x}, \tilde{t})$ be an optimal basic feasible solution to the LP relaxation $\max_{(x,t)\in\mathcal{P}} t$, which corresponds to a vertex of $\mathcal{P}$. We assume that $\tilde{x} \notin \mathcal{B}$, otherwise, $\tilde{x}$ is already an optimal solution to (4.6).

As $\tilde{z}$ is the point that we want to separate from $\mathrm{hypo}_{\mathcal{B}}(f)$, we follow the method presented in Sect. 3.4 to construct an intersection cut. According to [98, 159], we can use a feasible basis of the LP relaxation to create a simplicial cone $\mathcal{R}$. This cone $\mathcal{R}$ can be easily obtained from the simplex tableau associated with the chosen basis. In our case, we select the optimal basis defining $\tilde{z}$ so that $\tilde{z}$ is the apex of the corresponding cone $\mathcal{R}$. Moreover, we use EE_f as $\mathrm{hypo}_{\mathcal{B}}(f)$-free set. To determine whether the linear inequality (3.30) separates $\tilde{z}$ from $\mathrm{hypo}_{\mathcal{B}}(f)$, we need to verify whether $\tilde{z} \in \mathrm{int}(\mathsf{EE}_f)$.

The polyhedral outer approximation $\mathcal{P}$ gives rise to a piece-wise linear concave overestimating function of f over X: $\bar{f}(x) := \max_{(x,t)\in\mathcal{P}} t$, such that $\max_{(x,t)\in\mathcal{P}} t = \max_{x\in X} \bar{f}(x)$. This implies that $\tilde{t} = \bar{f}(\tilde{x})$. We then have the following observation.

Proposition 4.26. *Assume that f is not affine over X. If $\tilde{x} \in \mathrm{relint}(X)$, then $\bar{f}(\tilde{x}) > \bar{\mathsf{F}}_f(\tilde{x})$, i.e., $(\tilde{x}, \bar{f}(\tilde{x})) \in \mathrm{int}(\mathsf{EE}_f)$.*

Proof. As $\bar{f}$ is concave overestimator of f over X and $\bar{\mathsf{F}}_f$ is convex underestimator of f over X, $\bar{f} \geq \bar{\mathsf{F}}_f$ over X. Suppose, to aim at a contradiction, that $\bar{f}(\tilde{x}) = \bar{\mathsf{F}}_f(\tilde{x})$. Define a concave function $g := \bar{f} - \bar{\mathsf{F}}_f$, then for all $x \in X$, $g(x) \geq 0$, and $g(\tilde{x}) = 0$. By its concavity, there exists an affine overestimating function a of g, such that $g(\tilde{x}) = a(\tilde{x}) = 0$, and, for all $x \in X$, $0 \leq g(x) \leq a(x)$. As $\tilde{x} \in \mathrm{relint}(X)$, the affinity of a implies that $a = g = 0$ over X, i.e., $\bar{f} = \bar{\mathsf{F}}_f$ over X. So f is concave and convex over X and thus affine over X, which is a contradiction. □

The measure of the relative boundary $\mathrm{relbd}(X)$ is zero, so we can assume that a mild *relative interior condition* that $\tilde{x} \in \mathrm{relint}(X)$ holds with probability one. Under this assumption, the relaxation point $\tilde{z} = (\tilde{x}, \bar{f}(\tilde{x}))$ is in the relative interior of the extended envelope epigraph with probability one. This implies that the linear inequality (3.30) separates $\tilde{z}$ from $\mathrm{hypo}_{\mathcal{B}}(f)$ with probability one. In Sect. 6.4, we will empirically evaluate the effectiveness of these intersection cuts for various submodular maximization problems.

Throughout the rest of this chapter, we will encounter multiple nonconvex optimization problems. W.l.o.g., we will consider the simplicial cone defined by the simplex tableau associated with an optimal feasible basis of their LP relaxations. Such simplicial cones are commonly employed in computational implementations. For the sake of brevity, we refer to such cones as *optimal tableau cones*.

Extensions to SS functions

This section considers Boolean-hypograph and Boolean-superlevel sets for an SS function $f := f_1 - f_2$, where f_1 and f_2 are two submodular functions. We extend our previous results

on the Boolean-hypograph set of submodular functions; thus one can generate intersection cuts for a larger family of discrete nonconvex sets.

More specifically, we consider the following nonconvex set

$$\mathcal{S} := \{(x,t) \in \mathcal{B} \times \mathbb{R} : f(x) \geq \ell t\}, \tag{4.14}$$

with $\ell \in \{0,1\}$. Given a relaxation point $(\tilde{x},\tilde{t}) \notin \mathcal{S}$, we want to find cutting planes separating this point from $\mathcal{S}$.

Let $\bar{\mathsf{F}}_{f_1} := \max_{s\in\mathsf{EPM}_{f_1}} sx$ and $\bar{\mathsf{F}}_{f_2} := \max_{s\in\mathsf{EPM}_{f_2}} sx$ be extended envelopes of f_1, f_2, respectively. As $\bar{\mathsf{F}}_{f_1}$ (resp. $\bar{\mathsf{F}}_{f_2}$) is a convex extension of f_1 (resp. f_2), we have that $\mathcal{S} = \{(x,t) \in \mathcal{B} \times \mathbb{R} : \bar{\mathsf{F}}_{f_1}(x) - \bar{\mathsf{F}}_{f_2}(x) \geq \ell t\}$. By relaxing $\mathcal{B}$ to $\mathbb{R}^n$, a (nonconvex) continuous outer approximation of $\mathcal{S}$ is

$$\bar{\mathcal{S}} := \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \bar{\mathsf{F}}_{f_1}(x) - \bar{\mathsf{F}}_{f_2}(x) \geq \ell t\}. \tag{4.15}$$

Moreover, for all $x \in \mathcal{B}$, $(x,t) \in \bar{\mathcal{S}}$ if and only if $(x,t) \in \mathcal{S}$.

Special cases. When $\ell = 1$, $\mathcal{S}$ is the Boolean-hypograph of the SS function f; when $\ell = 0$, $\mathcal{S}$ is the 0-superlevel set of the SS function f over the Boolean hypercube. Setting $f_2 = 0$ and $\ell = 1$, the set $\mathcal{S}$ becomes the hypograph $\{(x,t) \in \mathcal{B} \times \mathbb{R} : f_1(x) \geq t\}$, which is studied in the previous section. Setting $f_1 = 0$, the relaxed set $\bar{\mathcal{S}}$ becomes $\{(x,t) \in \mathcal{B} \times \mathbb{R} : \bar{\mathsf{F}}_{f_2}(x) \leq -\ell t\}$. Let $(\tilde{x},\tilde{t}) \notin \bar{\mathcal{S}}$. Since $\bar{\mathsf{F}}_{f_2}(x) \geq \gamma^* x$ and $\bar{\mathsf{F}}_{f_2}(\tilde{x}) = \gamma^*\tilde{x}$ for any $\gamma^* \in \partial\bar{\mathsf{F}}_{f_2}(\tilde{x})$, then the simple outer approximation cut $\gamma^* x \leq -\ell t$ is a valid inequality for $\bar{\mathcal{S}}$ (hence for $\mathcal{S}$).

In general, we should separate intersection cuts specifically for SS functions. Let $\gamma^* \in \partial\bar{\mathsf{F}}_{f_2}(\tilde{x})$ be a solution to (4.11) associated with f_2, and we define the set

$$\mathcal{C}_{\tilde{x}} := \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \bar{\mathsf{F}}_{f_1}(x) - \gamma^* x \leq \ell t\}. \tag{4.16}$$

The following proposition characterizes $\mathcal{S}$-free sets for Eq. (4.15).

Proposition 4.27. *The set $\mathcal{C}_{\tilde{x}}$ in* (4.16) *is an $\mathcal{S}$-free set. Moreover, if $(\tilde{x},\tilde{t}) \notin \bar{\mathcal{S}}$, then $\mathcal{C}_{\tilde{x}}$ does not contain $\tilde{x}$ in its interior.*

Proof. We first prove that $\mathcal{C}_{\tilde{x}}$ is $\bar{\mathcal{S}}$-free. By definition, $\gamma^* x \leq \bar{\mathsf{F}}_{f_2}(x)$, which implies that $\bar{\mathsf{F}}_{f_1}(x) - \gamma^* x \geq \bar{\mathsf{F}}_{f_1}(x) - \bar{\mathsf{F}}_{f_2}(x)$. Therefore, for $(x,t) \in \operatorname{int}(\mathcal{C}_{\tilde{x}})$, we have that $\ell t > \bar{\mathsf{F}}_{f_1}(x) - \gamma^* x \geq \bar{\mathsf{F}}_{f_1}(x) - \bar{\mathsf{F}}_{f_2}(x)$, which implies that $(x,t) \notin \bar{\mathcal{S}}$. Hence, $\operatorname{int}(\mathcal{C}_{\tilde{x}}) \cap \bar{\mathcal{S}} = \varnothing$. Additionally, $\mathcal{C}_{\tilde{x}}$ is convex. These two facts imply that $\mathcal{C}_{\tilde{x}}$ is $\bar{\mathcal{S}}$-free. Since $\mathcal{S} \subseteq \bar{\mathcal{S}}$, $\mathcal{C}_{\tilde{x}}$ is also an $\mathcal{S}$-free set. Next, assume that $(\tilde{x},\tilde{t}) \notin \bar{\mathcal{S}}$, then $\ell\tilde{t} > \bar{\mathsf{F}}_{f_1}(\tilde{x}) - \bar{\mathsf{F}}_{f_2}(\tilde{x}) \leq \bar{\mathsf{F}}_{f_1}(\tilde{x}) - \gamma^*\tilde{x}$, so $(\tilde{x},\tilde{t}) \in \operatorname{int}(\mathcal{C}_{\tilde{x}})$. □

In [233, 271, 312], the authors study the sub/superlevel sets of some DC functions. Their construction of $\mathcal{S}$-free sets relies on a common *reverse-linearization technique*: reverse the set $\mathcal{S}$ by changing the sign of its defining inequality, and linearize one convex function.

In our case, f is an SS function, so we first need to extend the submodular and supermodular components of f. After the extension, we obtain a DC function. We can then apply the reverse-linearization technique to its continuous extension.

4.2 Convex envelopes of supermodular functions

We know overestimators for submodular functions in Sect. 3.2.2. However, we do not know their concave envelopes. In this section, w.l.o.g., we consider supermodular functions and their convex envelopes. We present a general characterization of convex envelopes of supermodular functions.

Let $f : Q \to \mathbb{R}$ be a supermodular function, where $Q := \{0,1\}^h$. We can use a bit representation to denote Boolean points in Q. For example, 10 denotes the point w that $w_1 = 1$ and $w_2 = 0$. For an affine function $a \cdot w + b$, its *supporting points* are the Boolean points in Q where $a \cdot w + b$ equals $f(w)$.

Lemma 4.28. *Every facet of f has $h+1$ affinely independent supporting points in Q.*

Proof of Lemma 4.7. Given $z \in \mathrm{dom}(f)$, $f(\rho z) = \rho^d f(z)$ is a real number for any $\rho \in \mathbb{R}_{++}$, so $\mathrm{int}(\mathrm{dom}(f))$ is a cone. Suppose that f is positive homogeneous of degree 1. For any $z \in \mathrm{dom}(f)$, $\Xi^f_{\check{z}}(z) = f(\check{z}) + \nabla f(\check{z}) \cdot (z - \check{z}) = \nabla f(\check{z}) \cdot z$, where the second equation follows from Euler's homogeneous function theorem: $f(\check{z}) = \nabla f(\check{z}) \cdot \check{z}$. For any $z = \rho\check{z}$ with $\rho \in \mathbb{R}_{++}$, $\Xi^f_{\check{z}}(z) = \nabla f(\check{z}) \cdot \rho\check{z} = \rho\Xi^f_{\check{z}}(\check{z}) = \rho f(\check{z}) = f(\rho\check{z})$, where the first and second equations follow from the previous result, the third follows from that $\Xi^f_{\check{z}}$ has the same value as f at $\check{z}$, and the last equation follows from the homogeneity. □

Proof. We note that any hyperplane in $\mathbb{R}^{h+1}$ is uniquely determined by $h+1$ affinely independent points. By definition, an affine underestimator $a \cdot w + b$ of f is a facet if and only if $a \cdot w + b \leq t$ is a facet of the epigraph $\mathrm{epi}_Q(f)$. Then, the affine underestimator is a facet, if and only if, there exists $h+1$ affinely independent points $\{(w^i, t^i)\}_{i \in [h+1]} \subseteq \mathrm{epi}_Q(f) \subseteq \mathbb{R}^{h+1}$ such that for all $i \in [h+1]$, $a \cdot w^i + b = t^i$. Moreover, t^i must equal $f(w^i)$, otherwise $a \cdot w^i + b = t^i < f(w^i)$, which implies that $a \cdot w + b \leq t$ does not underestimate f at w^i. Therefore, w^i is the supporting point of $a \cdot w + b$. Note that $\{(w^i, t^i)\}$ are affinely independent, if only if, $\{w^i\}_{i \in [h+1]}$ are affinely independent. □

We can enumerate all possible subsets of $h+1$ affinely independent points of Q. Each such subset $S := \{w^1, \ldots, w^{h+1}\}$ determines an function over $\mathbb{R}^h$ via the following affine combination:

$$f_S(w) := \{ \sum_{j \in [h+1]} \lambda_j f(w^j) : \exists \lambda \in \mathbb{R}^{h+1} \sum_{j \in [h+1]} \lambda_j = 1 \wedge \sum_{j \in [h+1]} \lambda_j w^j = w \}.$$

Due to the affine independence of S, the Barycentric coordinate λ for each w in the above affine combination is unique. We can consider f_S as a single-valued affine function and call it the *supported function* of S. Since, for all $w \in S$, $f_S(w) = f(w)$, solving the linear system $a \cdot w + b = f(w)(w \in S)$, we can compute a, b defining f_S. If the supported function f_S underestimates f, we call the subset S *facet-inducing*.

Assuming we have a collection of facets of f, we then determine whether these facets define the convex envelope of f. We recall that the convex hull of $h+1$ affinely independent points is called an h-*simplex*. A finite collection of h-simplices $\{P_k\}_k$ is called a *triangulation* of the unit cube $\mathcal{U}$, if the following conditions hold: (i) $\bigcup_k P_k = \mathcal{U}$, (ii) $P_k \cap P_{k'}$ is empty or a face of both $P_k \cap P_{k'}$ for all k, k', (iii) the vertices of P_k are contained in Q for all k.

Proposition 4.29. *Let $\{S_k\}_k$ be a collection of facet-inducing subsets of Q. For each k, let f_{S_k} be the supported function induced by S_k, and let $P_k := \mathrm{conv}(S_k)$ be the simplex spanned by S_k. If $\{P_k\}_k$ is a triangulation of $\mathcal{U}$, then $\mathrm{convenv}_{\mathcal{Q}}(f)(w) = \max_k f_{S_k}(w)$ for all $w \in \mathcal{U}$.*

Proof. Since $\bigcup_k P_k = \mathcal{U}$, for any $w \in Q$, there exists P_k such that $w \in P_k$. Since the vertices S_k of P_k are contained in Q, w must be in S_k. Therefore, $f_{S_k}(w) = f(w)$. This implies that $f(w) = \max_k f_{S_k}(w)$ for all $w \in Q$, i.e., $\max_k f_{S_k}$ is an exact convex underestimator. Suppose, to aim at a contradiction, that there exists another convex underestimator f' of f such that $f'(w) = f(w)$ for all $w \in Q$, and $f'(w') > \max_k f_{S_k}(w')$ for some $w' \in \mathcal{U}$. Again, since $\bigcup_k P_k = \mathcal{U}$, there exists P_k such that $w' \in P_k$, i.e., $w' \in \mathrm{conv}(S_k)$. Let $S_k = \{w^1, \dots, w^{h+1}\}$. It follows from $w' \in \mathrm{conv}(S_k)$ that there exists $\lambda \in [0,1]^{h+1}$ such that $\sum_{j\in[h+1]} \lambda_j = 1, \sum_{j\in[h+1]} \lambda_j w^j = w'$. Note that S_k induces the supported function f_{S_k}, which has supporting points S_k. This implies that $f_{S_k}(w') = \sum_{j\in[h+1]} \lambda_j f_{S_k}(w^j) = \sum_{j\in[h+1]} \lambda_j f(w^j)$. It follows from the convexity of f' that $f'(w') \leq \sum_{j\in[h+1]} \lambda_j f'(w^j) = \sum_{j\in[h+1]} \lambda_j f(w^j) \leq f_{S_k}(w') \leq \max_k f_{S_k}(w')$, which leads to a contradiction. Therefore, $\max_k f_{S_k}$ is the convex envelope of f over $\mathcal{Q}$. □

The collection $\{S_k\}_k$ is called *envelope-inducing* family in Q, if the presupposition "$\{P_k\}_k$ is a triangulation" in Prop. 4.29 is satisfied.

Chapter 5

Cutting planes for signomial programming

5.1 Introduction

In this chapter, we provide a deeper treatment of the *signomial term* $\psi_\alpha(x) := x^\alpha = \prod_{j\in[n]} x_j^{\alpha_j}$, where the exponent vector α is in $\mathbb{R}^n$, with respect to convexification and linearization within an sBB algorithm.

When all the terms in g in (3.12) are signomial terms, the problem (3.12) falls under the category of signomial programming (SP). In this scenario, we refer to (3.12) as the *natural formulation* of SP. The left-hand sides of the constraints in this formulation are referred to as *signomial functions*. The lifted set $\mathcal{S}_{\text{lift}}$ in the extended formulation (3.13) is called a *signomial lift*.

Since negative entries may be present in the exponent vector α, in general, variables of SP are assumed to be positive. We remark that the techniques in this chapter can also treat signomial terms in general mixed-integer NLP problems. The point of restriction on SP over positive variables is simply to make the theoretical treatment more readable and streamlined.

In the case of SP, LP relaxations can be derived from polyhedral outer approximations of the signomial lift in its extended formulation. A typical relaxation algorithm for SP involves factoring the signomial term $\psi_\alpha(x)$ into the product of n univariate signomial terms $x_i^{\alpha_i}$. Following the factorization, the algorithm proceeds to convexify and linearize the intermediate multilinear term and univariate functions. However, this factorable programming approach can lead to a weak LP relaxation and introduce additional auxiliary variables representing intermediate functions. These issues have been previously discussed in the context of pure multilinear terms [77, 110, 281].

We propose two cutting plane-based relaxation algorithms for SP. In contrast to the conventional factorable programming approach, our method employs a novel reformulation of the signomial lift. We study outer approximations of the following graph of the signomial term:

$$\mathcal{S} = \{(x, y) : y = x^\alpha\}. \tag{5.1}$$

We transform each nonlinear equality constraint $y_i = g_i(x) = x^{\alpha^i}$ in (3.14) to an equivalent constraint $\psi_\beta(u) - \psi_\gamma(v) = 0$, where $\beta > 0, \gamma > 0$, $\max(\|\beta\|_1, \|\gamma\|_1) = 1$, u, v are disjoint

sub-vectors of (x, y_i), and ψ_β, ψ_γ are concave functions. We consider approximating the following set

$$\mathcal{S}_{\rm st} \coloneqq \{(u, v) \in \mathbb{R}^{h+\ell}_+ : \psi_\beta(u) - \psi_\gamma(v) \leq 0\}. \tag{5.2}$$

Our first cutting plane algorithm is based on the intersection cut paradigm [102]. One can approximate a nonconvex set $\mathcal{S}$ using its simplicial conic outer approximation. This requires the construction of $\mathcal{S}$-free sets, which are closed convex sets containing none of the interiors of $\mathcal{S}$. The main insight about $\mathcal{S}$-free sets for a nonconvex set $\mathcal{S}$ is that they provide an explicit and useful description of convex parts of the infeasible space w.r.t. $\mathcal{S}$.

Our second cutting plane algorithm is based on the conventional underestimating and overestimating techniques. To ensure the convergence of the sBB algorithm, a common assumption for SP is that all variables are bounded. We generate linear underestiamtor (resp. linear overestimator) for $\psi_\beta(u)$ (resp. $\psi_\gamma(v)$). This yields a convex outer approximation of $\mathcal{S}_{\rm st}$.

5.1.1 Literature review

SP finds applications in diverse areas, such as aeronautics [243], chemical reactors [69], delta-sigma modulator topologies [173], design of heat exchanger networks [66], inductors [181], and trim-loss minimization [173].

The majority of relaxations for SP are derived from its generalized geometric programming (GPP) formulation, which is an exponential transformation [130] of its natural formulation. The exponential transformation replaces positive variables x by exponentials $\exp(z)$, where z are real variables. The authors of [221] show that signomial functions in GGP are difference-of-convex (DC) functions. For the signomial function in each constraint of GGP, they construct linear underestimators of its concave part; the author of [275] constructs linear underestimators of the whole function via the mean value theorem. The author of [310] proposes inner approximations of GGP via the inequality of arithmetic and geometric means (AM-GM inequality). The authors of [82, 127, 235] construct non-negativity certificates for signomial functions via the AM-GM inequality, and propose a hierarchy of convex relaxations for GGP. Exponential transformations can be combined with other variable transformations, such as power transformations, and the inverse transformations can be approximated by piece-wise linear functions, see [207, 218, 219].

The solvers SCIP [62], `BARON` [284], `ANTIGONE` [227], and `MISO` [226, 228] are capable of solving the natural formulation of SP or its extended formulation within a global ϵ-optimality using the sBB algorithm. Specifically, `MISO` is a specialized solver for SP that employs exponential transformations of some signomial terms only when necessary. Due to the following reasons, exponential transformations can complicate general-purpose solvers. First, in certain NLP problems, signomial terms may appear only as a subset of the nonlinear terms of $g(x)$. In such cases, solvers may need to enforce the inverse transformation $x_j = \ln(z_j)$, which requires additional processing for convexification algorithms. Secondly, when dealing with mixed-integer SP, if some variables of x are integers, exponential transformations lead to certain components of z becoming discrete but not necessarily integer. As a result, the sBB algorithm needs to adapt its branching rules.

While considerable attention has been devoted to constructing relaxations for GGP, the literature on relaxations for the extended natural formulation of SP is relatively limited. The convex relaxations employed in the mentioned solvers mainly rely on factorable programming

[204, 223]. Since exponential transformations are nonlinear variable transformations, it is impossible to directly apply the relaxations designed for the GGP formulation to the natural formulation.

Numerous research efforts have been dedicated to improving the relaxation techniques for multilinear terms and univariate/bivariate functions commonly used in factorable programming [35]. Multilinear terms over the unit hypercube are vertex polyhedral, and their envelopes over the unit hypercube admit simple extended formulations [262]. Notably, closed forms for the convex envelopes of bilinear functions [11, 223] and trilinear functions [224, 225] over hypercubes are well-established. In [276], the author presents convex envelopes for multilinear functions (sum of multilinear terms) over the unit hypercube and specific discrete sets. For a comprehensive analysis of multilinear term factorization via bilinear terms, we refer to [217, 281]. Additionally, [77] offers an in-depth examination of quadrilinear function factorization through bilinear and trilinear terms, while [110] presents a computational study on extended formulations.

Convexifying univariate/bivariate functions plays an important role in the field of MINLP optimzation. In [205], convex envelopes for monomials with odd degrees are derived. An approach presented in [211] enables the evaluation of the convex envelope of a bivariate function over a polytope and separating its supporting hyperplane by solving low-dimensional convex optimization problems. The convex optimization problems are further reduced by solving a Karush-Kuhn-Tucker system [210]. In [209], convex envelopes for bilinear, fractional, and other bivariate functions over a polytope are constructed using a polyhedral subdivision technique. Additionally, [239, 282] employ polyhedral subdivision and lift-project methods to derive explicit forms of convex envelopes for various non-convex functions, including a specific subclass of bivariate signomial terms.

Convexifying high-order multivariate functions poses a significant challenge, and the available literature on convex underestimators for trivariate functions is relatively scarce. In [175, 176], the authors propose a novel framework for relaxing composite functions in nonlinear programs. Another approach involves using the intersection cut paradigm [102] to approximate nonconvex functions. This paradigm can generate cutting planes to strengthen LP relaxations of NLP problems. Constructing intersection cuts involves searching for an $\mathcal{S}$-free set, where $\mathcal{S}$ represents a nonconvex set defined by nonconvex functions.

The study of intersection cuts originated in the context of NLP [292]. Gomory later introduced the concept of corner polyhedron [159], and intersection cuts were explored in the field of integer programming [31]. The modern definition of intersection cuts for arbitrary sets $\mathcal{S}$ is from [125, 156]. For more comprehensive details, we refer to [16, 39, 40, 109, 119, 125, 261]. Recent research has revealed $\mathcal{S}$-free sets for various nonconvex sets encountered in structured NLP problems. Examples include outer product sets [63], sublevel sets of DC functions [271], quadratic sets [233], and graphs of bilinear terms [138]. Intersection cuts have also been developed for convex mixed-integer NLP problems [17, 46, 194, 195, 230] and for bilevel programming [137].

5.1.2 Contribution

We give the transformation procedure leading to $\mathcal{S}_{\mathrm{st}}$ and construct $\mathcal{S}_{\mathrm{st}}$-free sets from the transformation. We show that these sets are also signomial-lift-free and maximal in the non-negative orthant. We also discuss the separation of intersection cuts.

Our second cutting plane algorithm aims at approximating $\mathcal{S}_{\mathrm{st}}$ within a hypercube. In Sect. 5.3, we provide an extended formulation for the convex envelope of the concave function ψ_β over the hypercube. This formulation yields a convex outer approximation of $\mathcal{S}_{\mathrm{st}}$, allowing us to generate outer approximation cuts through projection. We prove that ψ_β is a supermodular function. For $h = 2$, we provide a closed-form expression for its convex envelope by exploiting supermodularity: this allows us to remove the projection step.

About the computational part of this study, we note that signomials are one of the four main types of nonlinearity occurring in the mixed-integer NLP library (MINLPLib) [47, 75]. Our relaxation approach does not require factorization or introduce intermediate functions, making the implementation of the proposed cutting planes within the general-purpose solver SCIP straightforward.

5.1.3 Notation

For a vector $x \in \mathbb{R}^n$, given $J \subseteq [n]$, $x_J = (x_j)_{j\in J}$ denotes the sub-vector formed by entries indexed by J. Given a differentiable function f, for a $\tilde{x} \in \mathrm{dom}(f)$, $\nabla f(\tilde{x})$ denotes the gradient of f at $\tilde{x}$ and

$$\Xi^f_{\tilde{x}}(x) := f(\tilde{x}) + \nabla f(\tilde{x}) \cdot (x - \tilde{x}). \tag{5.3}$$

The word *linearization* has two different meanings in mathematical programming: (i) a symbolic one, which entails the replacement of all occurrences of nonlinear term $\tau(x)$ with a new variable t, followed by the addition of a new constraint $t = \tau(x)$ in a formulation — this is also called a "lifting"; (ii) an analytic one, which involves the replacement of a convex nonlinear constraint with an affine subspace tangent to the nonlinear surface at a given point. All mentions of linearization in the rest of this chapter refer to the second meaning.

5.1.4 Outline of the chapter

In Sect. 5.2, we construct several families of $\mathcal{S}$-free sets, study their maximalities, and derive intersection cuts. In Sect. 5.3, we construct a nonlinear relaxation of the signomial term set, and derive outer approximation cuts through projection. In Sect. 5.4, we perform computational tests on instances from MINLPLib and observe improvements to SCIP's default settings due to the proposed valid inequalities.

5.2 Signomial-lift-free sets and intersection cuts

In this section, we construct (maximal) signomial-lift-free sets and generate intersection cuts for SP.

5.2.1 Signomial-lift-free and signomial-term-free sets

We introduce and study formulations of signomial term sets. We transform signomial term sets into DCC sets. Furthermore, we construct signomial-term-free sets and lift them to signomial-lift-free sets. The maximality of these sets is investigated, and a comparison is carried out between signomial-term-free sets derived from different DCC formulations.

We consider an n-variate signomial term $\psi_\alpha(x)$ arising in the extended formulation (3.13) of SP. The exponent vector α may contain negative/zero/positive entries. We extract two sub-vectors α^- and α^+ from α such that $\alpha^- < 0 \in \mathbb{R}^{h'}$ and $\alpha^+ > 0 \in \mathbb{R}^{\ell'}$, and let $x_- \in \mathbb{R}^{h'}$ and $x_+ \in \mathbb{R}^{\ell'}$ be the corresponding sub-vectors of x. Entries x_j with $\alpha_j = 0$ are excluded from consideration, and so $h' + \ell'$ may be smaller than n. Since $\psi_\alpha(x)$ only depends on x_- and x_+, it can be represented in the form of $x_-^{\alpha^-} x_+^{\alpha^+}$ of lower order. Then, a *signomial term set* is defined as epigraph or hypograph of $x_-^{\alpha^-} x_+^{\alpha^+}$:

$$\mathcal{S}_{\rm st} = \{(x_-, x_+, t) \in \mathbb{R}^{h'+\ell'+1}_{++} : t \lesseqgtr x_-^{\alpha^-} x_+^{\alpha^+}\}. \tag{5.4}$$

We first give DCC reformulations of signomial term sets. Let $\lesseqgtr$ denote $<$ or $>$. The interior of $\mathcal{S}_{\rm st}$ in (5.4) is

$$\operatorname{int}(\mathcal{S}_{\rm st}) = \{(x_-, x_+, t) \in \mathbb{R}^{h'+\ell'+1}_{++} : t \lessgtr x_-^{\alpha^-} x_+^{\alpha^+}\}.$$

Reorganizing the signomial terms and taking the closure of the set, we recover

$$\mathcal{S}_{\rm st} = \{(x_-, x_+, t) \in \mathbb{R}^{h'+\ell'+1}_{+} : t x_-^{-\alpha^-} \lesseqgtr x_+^{\alpha^+}\}.$$

Notably, the exponents associated with signomial terms on both sides are now strictly positive. Let $\mathbb{R}_{++}$ denote the positive orthant. Let $u \coloneqq (t, x_-), v \coloneqq x_+$, let $h \coloneqq h' + 1$, and let $\ell \coloneqq \ell'$. Then, $\psi_{\beta'}(u) = t x_-^{-\alpha^-}$ and $\psi_{\gamma'}(v) = x_+^{\alpha^+}$, where $\beta' \coloneqq (1, -\alpha^-) \in \mathbb{R}^h_{++}$ and $\gamma' \coloneqq \alpha^+ \in \mathbb{R}^\ell_{++}$. After the change of variables, the set admits the following form:

$$\mathcal{S}_{\rm st} = \{(u, v) \in \mathbb{R}^{h+\ell}_{+} : \psi_{\beta'}(u) \lesseqgtr \psi_{\gamma'}(v)\}, \tag{5.5}$$

where $\lesseqgtr$ denote $\leq$ or $\geq$.

The formulation (5.5) exhibits symmetry between u and v. We can therefore consider w.l.o.g. the inequality "$\leq$" throughout the subsequent analysis. Since the signomial terms $\psi_{\beta'}(u), \psi_{\gamma'}(v)$ are non-negative over $\mathbb{R}^h_+, \mathbb{R}^\ell_+$, we can take any positive power $\eta \in \mathbb{R}_{++}$ on both sides of (5.5). Finally, the signomial term set in (5.4) admits the following form:

$$\mathcal{S}_{\rm st} = \{(u, v) \in \mathbb{R}^{h+\ell}_{+} : \psi_{\beta}(u) - \psi_{\gamma}(v) \leq 0\}, \tag{5.6}$$

where $\beta \coloneqq \eta\beta'$, and $\gamma \coloneqq \eta\gamma'$.

A signomial term $\psi_\alpha(x)$ is said to be a *power function* if $\alpha \geq 0$, and $\|\alpha\|_1 \leq 1$. According to [18, 84], power functions are concave over the non-negative orthant; if additionally $\|\alpha\|_1 = 1$, $\psi_\alpha(x)$ is positive homogeneous of degree 1. Through an appropriate scaling of the parameter η, we obtain a family of DCC reformulations (5.6) of signomial term sets. We let $\mathcal{G} \coloneqq \mathbb{R}^{h+\ell}_+$, and use the reverse-linearization technique to construct signomial-term-free sets. We recall that the definition of the operator Ξ is given in Eq. (5.3).

Proposition 5.1. *Let* $\max(\|\beta\|_1, \|\gamma\|_1) \leq 1$. *For any* $\tilde{v} \in \mathbb{R}^\ell_{++}$,

$$\mathcal{C} \coloneqq \{(u, v) \in \mathbb{R}^h_+ \times \mathbb{R}^\ell : \psi_\beta(u) - \Xi^{\psi_\gamma}_{\tilde{v}}(v) \geq 0\} \tag{5.7}$$

is a signomial-term-free ($\mathcal{S}_{\rm st}$-free) set. If $\max(\|\beta\|_1, \|\gamma\|_1) = 1$, *then $\mathcal{C}$ is a maximal signomial-term-free set in $\mathcal{G}$.*

Proof. Since $\max(\|\beta\|_1, \|\gamma\|_1) \leq 1$, $\psi_\beta(u), \psi_\gamma(v)$ are concave. By Lemma 3.22, $\mathcal{C}$ is signomial-term-free. If $\max(\|\beta\|_1, \|\gamma\|_1) = 1$, then at least one of $\|\beta\|_1, \|\gamma\|_1$ is 1. Therefore, one of $\psi_\beta(u), \psi_\gamma(v)$ is positive homogeneous of degree 1. Moreover, $\psi_\beta(u), \psi_\gamma(v)$ are both continuously differentiable and positive over positive orthants $\mathbb{R}^h_{++}, \mathbb{R}^\ell_{++}$ (the interiors of their domains). Since $\mathcal{G} = \mathrm{dom}(\psi_\beta) \times \mathrm{dom}(\psi_\gamma)$, by Thm. 4.8, $\mathcal{C} \cap \mathcal{G} = \{(u, v) \in \mathcal{G} : \psi_\beta(u) - \Xi^{\psi_\gamma}_{\tilde{v}}(v) \geq 0\}$ is a maximal signomial-term-free set in $\mathcal{G}$. Therefore, $\mathcal{C}$ is also a maximal signomial-term-free set in $\mathcal{G}$. □

Given that $\max(\|\beta\|_1, \|\gamma\|_1) = 1$ results in a desirable DDC formulation for the signomial term set, we refer to this formulation as its *normalized DCC formulation*. Comparing Prop. 5.1 to Thm. 4.8, we extend the domain of $\Xi^{\psi_\gamma}_{\tilde{v}}(v)$ from $\mathbb{R}^\ell_+$ to $\mathbb{R}^\ell$, since it is an affine function. However, the further extension requires a non-trivial concave extension of the power function ψ_β, for which we are unaware of a closed-form expression.

We have reduced the n-variate signomial term $\psi_\alpha(x)$ to a signomial term $x_-^{\alpha^-} x_+^{\alpha^+}$ of lower order and constructed the corresponding signomial-term-free sets. A similar reduction is observed for g_i to g'_i in Sect. 4.1.1, where we demonstrate the relationship between $\mathcal{S}_{\mathrm{lift}}$-free sets and $\mathrm{epi}(g'_i)$-free/$\mathrm{hypo}(g'_i)$-free sets.

Next, we let the lifted set $\mathcal{S}_{\mathrm{lift}}$ be the signomial lift, where all g_i are signomial terms. Each equality constraint $y_i = g_i(x)$ defining the signomial lift is equivalent to two inequality constraints $y_i \lessgtr g_i(x)$. Applying the normalized DDC reformulation to these inequality constraints, we thus obtain a reformulation of the signomial lift, which we call its *normalized DCC reformulation*.

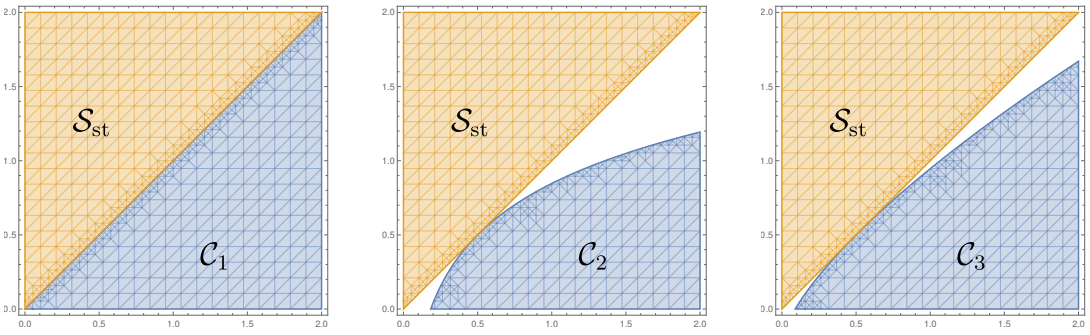
Corollary 5.2. *Let $\mathcal{C}$ be as in* (5.7)*, where $\psi_\alpha = g_i$ for some $i \in [k]$ and $\max(\|\beta\|_1, \|\gamma\|_1) = 1$. Then the orthogonal lifting of $\mathcal{C}$ w.r.t. g_i is a maximal signomial-lift-free ($\mathcal{S}_{\mathrm{lift}}$-free) set in the non-negative orthant.*

Proof. We verify that the presuppositions of Thm. 4.6 are satisfied by the signomial lift. For every $i \in [k]$, the signomial term g_i is continuous, and its domain and range are $\mathbb{R}_{++}$. Let J_i be the index set of variables of its reduced signomial term g'_i. Let $\mathcal{X} \coloneqq \bigtimes_{j\in[n]} \mathbb{R}_{++}, \mathcal{Y} \coloneqq \bigtimes_{j\in[k]} \mathbb{R}_{++}$. For all $j \in [n+k]$, let $\mathcal{D}_j \coloneqq \mathbb{R}_{++}$. For all $i \in [k]$, let $\mathcal{X}^i \coloneqq \bigtimes_{j\in J_i} \mathbb{R}_{++}, \mathcal{Y}^i \coloneqq \mathbb{R}_{++}$. The underlying sets of the signomial lift are $\mathcal{X}, \mathcal{Y}, \{\mathcal{X}^i, \mathcal{Y}^i\}_{i\in[k]}$, which are 1d-convex decomposable by $\{\mathcal{D}_j\}_{j\in[n+k]}$. By Prop. 5.1, $\mathcal{C}$ is a maximal $\mathrm{hypo}(g'_i)$-free set in $\mathcal{X}^i \times \mathcal{Y}^i$. By Thm. 4.6, its orthogonal lifting w.r.t. g_i is a maximal signomial-lift-free set in positive orthant. By continuity of $\psi_\beta(u), \psi_\gamma(v)$, we can change the ground set from the positive orthant to its closure, i.e., non-negative orthant. □

We next give some examples of signomial-term-free sets from different DDC formulations.

Example 5.3 (Comparison of DCC formulations)**.** *Consider $\mathcal{S}_{\mathrm{st}} = \{(u, v) \in \mathbb{R}^2_+ : u \leq v\}$, which is already in normalized DCC formulation. It is easy to see that $\mathcal{C}_1 \coloneqq \{(u, v) \in \mathbb{R}_+ \times \mathbb{R} : u \geq v\}$ is a maximal $\mathcal{S}_{\mathrm{st}}$-free set in $\mathbb{R}^2_+$ given by Prop. 5.1. Let $\tilde{v} \in \mathbb{R}_{++}$ be a linearization point. As $\mathrm{int}(\mathcal{S}_{\mathrm{st}})$ is equivalent to the logarithmic reformulation $\{(u, v) \in \mathbb{R}^2_+ : \log(u) \leq \log(v)\}$, which is also a DCC formulation, applying the reverse-linearization technique at $\tilde{v}$ yields $\mathcal{C}_2 \coloneqq \{(u, v) \in \mathbb{R}^2_+ : \log(u) - (\log(\tilde{v}) + (v - \tilde{v})/\tilde{v}) \geq 0\}$, which is also an $\mathcal{S}_{\mathrm{st}}$-free set. For any $0 < \eta < 1$, $\mathcal{S}_{\mathrm{st}} = \{(u, v) \in \mathbb{R}^2_+ : u^\eta \leq v^\eta\}$ is a DDC set, applying the reverse-linearization technique at $\tilde{v}$ yields $\mathcal{C}_3 \coloneqq \{(u, v) \in \mathbb{R}^2_+ : u^\eta - ((1-\eta)\tilde{v}^\eta + \eta\tilde{v}^{\eta-1}v) \geq 0\}$, which is also an $\mathcal{S}_{\mathrm{st}}$-free set. However, $\mathcal{C}_2, \mathcal{C}_3$ cannot be maximal in $\mathbb{R}^2_+$, because their intersections with $\mathbb{R}^2_+$*

are not polyhedral. These sets are visualized in Fig. 5.1 with a linearization point $\tilde{v} = 0.5$ *and scaling parameter* $\eta = 0.7$.

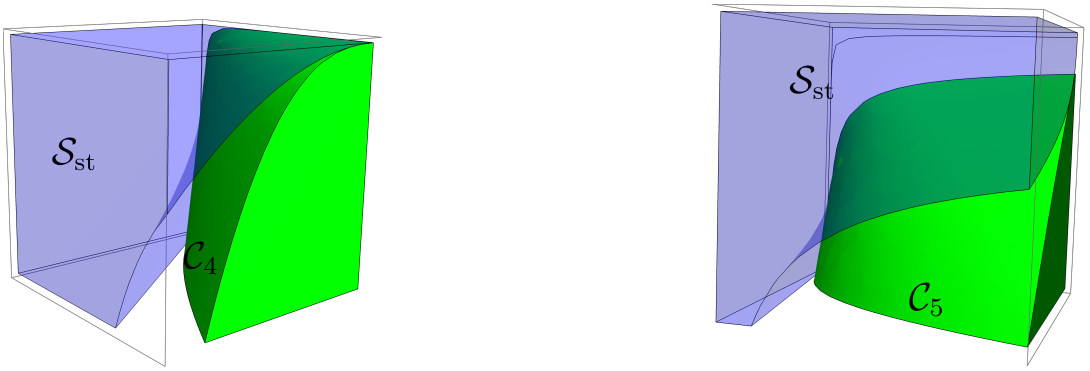


(a) $\mathcal{S}_{\text{st}}$ and $\mathcal{C}_1$. (b) $\mathcal{S}_{\text{st}}$ and $\mathcal{C}_2$. (c) $\mathcal{S}_{\text{st}}$ and $\mathcal{C}_3$.

Figure 5.1 $\mathcal{S}_{\text{st}}$-free sets.

Example 5.4. *Consider the hypograph of signomial term* $x_1^{-2}x_2^2$ *and* $\mathcal{S}_{\text{st}} = \{(x, y) \in \mathbb{R}^3_+ : y \leq x_1^{-2}x_2^2\}$. *For* $(x, y) \in \mathbb{R}^3_{++}$, $y \leq x_1^{-2}x_2^2$ *if and only if* $y^{1/3}x_1^{2/3} \leq x_2^{2/3}$. *The following set is maximal* $\mathcal{S}_{\text{st}}$*-free in* $\mathcal{G} = \mathbb{R}^3_+$*:* $\mathcal{C}_4 := \{(x, y) \in \mathbb{R}^3_+ : y^{1/3}x_1^{2/3} \geq \tilde{x}_2^{2/3} + \frac{2}{3}\tilde{x}_2^{-1/3}(x_2 - \tilde{x}_2)\}$, *where* $\tilde{x}_2 \in \mathbb{R}_{++}$. *See Fig. 5.2a for* $\tilde{x}_2 = 0.2$.

Example 5.5. *Consider the epigraph of signomial term* $x_1^3x_2$ *and* $\mathcal{S}_{\text{st}} = \{(x, y) \in \mathbb{R}^3_+ : y \geq x_1^3x_2\}$. *For* $(x, y) \in \mathbb{R}^3_{++}$, $y \geq x_1^3x_2$ *if and only if* $y^{1/4} \geq x_1^{3/4}x_2^{1/4}$. *The following set is maximal* $\mathcal{S}_{\text{st}}$*-free in* $\mathcal{G} = \mathbb{R}^3_+$*:* $\mathcal{C}_5 := \{(x, y) \in \mathbb{R}^3_+ : \tilde{y}^{1/4} + \frac{1}{4}\tilde{y}^{-3/4}(y - \tilde{y}) \leq x_1^{3/4}x_2^{1/4}\}$, *where* $\tilde{y} \in \mathbb{R}_{++}$. *See Fig. 5.2b for* $\tilde{y} = 0.2$.



(a) $\mathcal{S}_{\text{st}}$ and $\mathcal{C}_4$. (b) $\mathcal{S}_{\text{st}}$ and $\mathcal{C}_5$.

Figure 5.2 Two examples of $\mathcal{S}_{\text{st}}$ and $\mathcal{S}_{\text{st}}$-free sets.

5.2.2 Computing intersection cuts

We focus on the separation of intersection cuts for the extended formulation of SP. In Sect. 3.4, we presented a method to construct a simplicial cone $\mathcal{R}$ from an LP relaxation. The vertex of that cone is a relaxation solution $\tilde{z} = (\tilde{x}, \tilde{y})$.

We assume that the LP relaxation includes all the linear constraints from (3.12). If $\tilde{z}$ is not feasible for (3.12), then $\tilde{z}$ does not belong to the signomial lift. Hence, there exists a signomial

term g_i such that $\tilde{y}_i \neq g_i(\tilde{x})$. Given the reduced form g_i', we obtain a signomial term set $\mathcal{S}_{\rm st}$: if $g_i(\tilde{x}) > \tilde{y}_i$, we choose $\mathcal{S}_{\rm st}$ as the epigraph of g_i'; otherwise, we select it as the hypograph of g_i'. This signomial term set yields a signomial-term-free set $\mathcal{C}$ in (5.7) containing $(\tilde{u}, \tilde{v})$ in its interior (Lemma 3.22). By applying the orthogonal lifting, we can transform $\mathcal{C}$ into a signomial-lift-free set $\bar{\mathcal{C}}$ as stated in Cor. 5.2.

We next show how to construct an intersection cut in (3.30). It suffices to compute step lengths η_j^* in (3.29) along extreme rays r^j of $\mathcal{R}$. Each step length η_j^* corresponds to a boundary point $\tilde{z}+\eta_j^* r^j$ in $\mathrm{bd}(\bar{\mathcal{C}})$. The left-hand-side $\psi_\beta(u) - \Xi_{\tilde{v}}^{\psi_\gamma}(v)$ of the inequality in (5.7) is a concave function over $(u,v) \in \mathbb{R}_+^h \times \mathbb{R}^\ell$. Its restriction along the ray $\tilde{z} + \eta_j r^j$ $(\eta_j \in \mathbb{R}_+)$ is a univariate concave function:

$$\tau_j : \mathbb{R}_+ \to \mathbb{R}, \eta_j \mapsto \tau_j(\eta_j) \coloneqq \psi_\beta(\tilde{u} + r_u^j \eta_j) - \Xi_{\tilde{v}}^{\psi_\gamma}(\tilde{v} + r_v^j \eta_j),$$

where r_u^j and r_v^j are the projections of r^j on u and v respectively. Let $\bar{\eta}_j \coloneqq \sup_{\eta_j \geq 0}\{\eta_j : \tilde{u} + r_u^j \eta_j \geq 0\}$. Therefore, η_j^* is the first point in $[0, \bar{\eta}_j]$ satisfying the boundary condition: either $\tau_j(\eta_j^*) = 0$ or $\eta_j^* = \bar{\eta}_j$. Since τ_j is a univariate concave function and $\tau_j(0) > 0$, there is at most one positive point in $\mathbb{R}_+$ where τ_j is zero. We employ the bisection search method [255] to find such η_j^*.

5.3 Convex outer approximation

In this section, we propose a convex nonlinear relaxation for the extended formulation (3.13) of SP. This relaxation allows us to generate valid linear inequalities, known as outer approximation cuts, for SP. Unlike intersection cuts, outer approximation cuts do not require an LP relaxation *a priori* .

Our goal is to construct a convex outer approximation of the signomial lift. This gives rise to the convex nonlinear relaxation of SP. To ensure the convergence of the sBB algorithm, the feasible region of the extended formulation (3.13) should be compact. Therefore, we assume that the signomial lift is in a hypercube.

Our algorithm approximates every signomial term set in the hypercube generated from the signomial lift. W.l.o.g., we consider a signomial term set in hypergraph or epigraph form. As Sect. 5.2.1, we can convert it in normalized DDC formulation:

$$\mathcal{S}_{\rm st} = \{(u,v) \in \mathcal{U} \times \mathcal{V} : \psi_\beta(u) - \psi_\gamma(v) \leq 0\}, \tag{5.8}$$

where $\max(\|\beta\|_1, \|\gamma\|_1) = 1$, and $\mathcal{U}, \mathcal{V}$ are two hypercubes in $\mathbb{R}_+^h, \mathbb{R}_+^\ell$ respectively. The signomial term set is generally nonconvex, so we should find a convex outer approximation of $\mathcal{S}_{\rm st}$.

Our construction involves convexifying the concave function ψ_β in (5.8). To do so, we will use the formal concepts of convex underestimators and convex envelopes.

Given a function f and a closed set $\mathcal{D} \subseteq \mathbb{R}^p$, a convex function $f' : \mathrm{conv}(\mathcal{D}) \to \mathbb{R}$ is said to be a convex underestimator of f over $\mathcal{D}$, if, for all $x \in \mathcal{D}$, $f'(x) \leq f(x)$. The convex envelope $\mathrm{convenv}_{\mathcal{D}}(f)$ of f is defined as the point-wise maximum convex underestimator of f over D, i.e., $\mathrm{epi}(\mathrm{convenv}_{\mathcal{D}}(f)) = \mathrm{conv}(\mathrm{epi}_{\mathcal{D}}(f))$, where $\mathrm{epi}_{\mathcal{D}}(f) \coloneqq \{(x,t) \in \mathcal{D} \times \mathbb{R} : f(x) \leq t\}$.

The following lemma gives an extended formulation of the convex envelope of a concave function over a polytope, where the formulation is uniquely determined by the function values

at the vertices of the polytope. Based on Lemma 3.5, we observe that the concave function f is convex-extensible from its vertices (i.e., $\mathrm{convenv}_P(f)(x) = \mathrm{convenv}_Q(f)(x)$ for $x \in P$), and $\mathrm{convenv}_P(f)$ is a polyhedral function.

For the case of $P = \mathcal{U} \coloneqq \prod_{j\in[h]}[\underline{u}_j, \overline{u}_j]$ and $f = \psi_\beta$, $Q = \{q \in \mathbb{R}^h : \forall j \in [h]\ q_j = \underline{u}_j \vee q_j = \overline{u}_j\}$ is the set of vertices of the hypercube $\mathcal{U}$. This yields an extended formulation of $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)$. Replacing ψ_β by its convex envelope $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)$, we obtain a convex outer approximation of $\mathcal{S}_{\mathrm{st}}$ in (5.8):

$$\overline{\mathcal{S}}_{\mathrm{st}} \coloneqq \{(u, v) \in \mathcal{U} \times \mathcal{V} : \mathrm{convenv}_{\mathcal{U}}(\psi_\beta)(u) \leq \psi_\gamma(v)\}.$$

By using this extended formulation, our convex nonlinear relaxation of SP incorporates additional auxiliary variables. Specifically, we require 2^h variables λ_q to represent each convex envelope. For most SP problems in the MINLPLib, where the degrees of signomial terms are less than 6, and h is less than 3, the convex outer approximation remains computationally feasible.

5.3.1 Outer approximation cuts

To enhance efficiency, we propose a cutting plane algorithm to separate valid linear inequalities in the (u, v)-space from the extended formulation of the convex outer approximation. This algorithm generates a low-dimensional projection of $\overline{\mathcal{S}}_{\mathrm{st}}$.

Given a point $(\tilde{u}, \tilde{v}) \in \mathcal{U} \times \mathcal{V}$, the algorithm determines whether it belongs to $\overline{\mathcal{S}}_{\mathrm{st}}$. This verification can be done by checking the sign of $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)(\tilde{u}) - \psi_\gamma(\tilde{v})$. If $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)(\tilde{u}) - \psi_\gamma(\tilde{v}) \leq 0$, then $(\tilde{u}, \tilde{v}) \in \overline{\mathcal{S}}_{\mathrm{st}}$.

Since $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)$ is a convex polyhedral function, our cutting plane algorithm evaluates the function by searching for an affine underestimator $a \cdot u + b$ of $\mathrm{convenv}_{\mathcal{U}}(u)$ such that $a \cdot \tilde{u} + b = \mathrm{convenv}_{\mathcal{U}}(\tilde{u})$. If $(\tilde{u}, \tilde{v}) \notin \overline{\mathcal{S}}_{\mathrm{st}}$, then $a \cdot u + b \leq \psi_\gamma(\tilde{v})$ is a valid nonlinear inequality of $\overline{\mathcal{S}}_{\mathrm{st}}$. Consequently, our cutting plane algorithm linearizes the inequality, resulting in an outer approximation cut $a \cdot u + b \leq \Xi_{\tilde{v}}^{\psi_\gamma}(v)$: we recall that Ξ is defined in Eq. (5.3).

Due to Lemma 3.5, we can solve the following LP to find the affine underestimator:

$$\max_{a\in\mathbb{R}^h, b\in\mathbb{R}} a \cdot \tilde{u} + b \quad \text{s.t.}\ \forall q \in Q\ a \cdot q + b \leq \psi_\gamma(q), \tag{5.9}$$

where we omit the linear constraints that bound (a, b). The maximum value obtained from this LP is exactly $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)(\tilde{u})$. The affine underestimator $a \cdot \tilde{u} + b$ is called a *facet* of the envelope $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)$, if $a \cdot \tilde{u} + b \leq t$ is a facet of $\mathrm{epi}(\mathrm{convenv}_{\mathcal{U}}(\psi_\beta))$. It should be noted that the solution of the LP is not necessarily a facet.

For $h = 1, 2$, we can explicitly provide projected formulations of convex envelopes of power functions. This enables us to obtain facets of $\mathrm{convenv}_{\mathcal{U}}(\psi_\beta)$ without the need to solve LPs. As a result, our cutting plane algorithm can efficiently separate outer approximation cuts for low-order problems.

To simplify our presentation, we translate and scale the domain of ψ_β to $[0, 1]^h$. This yields a new function $s(w) \coloneqq \psi_\beta(u)$, where for all $j \in [h]$, $u_j \coloneqq \overline{u}_j + (\overline{u}_j - \underline{u}_j)w_j$. After these transformations, we have $\mathcal{U} = [0, 1]^h$ and $Q = \{0, 1\}^h$. W.l.o.g., we focus on studying and computing facets of $\mathrm{convenv}_{\mathcal{U}}(s)$. For $h = 1$, the only facet is $s(0) + (s(1) - s(0))w_1$.

A set $D \subseteq \mathbb{R}^h$ is called a *product set*, if $D = \bigtimes_{j\in[h]} D_j$ for $D_j \subseteq \mathbb{R}$. Let D be a product set. A function $f : D \to \mathbb{R}$ is *supermodular* over D (Section 2.6.1 of [287]), if the increasing difference condition holds: for all $w^1, w^2 \in D, d \in \mathbb{R}^h_+$ such that $w^1 \leq w^2$ and $w^1+d, w^2+d \in D$, $f(w^1 + d) - f(w^1) \leq f(w^2 + d) - f(w^2)$. The following operations preserve supermodularity.

Lemma 5.6. *Let $w' \in \mathbb{R}^h, \rho \in \mathbb{R}^h_{++}$, and let D' be a product subset of D. The following results hold: (restriction) f is supermodular over D';(translation) $f(w + w')$ is supermodular over $D - d$; (scaling) $f(\rho * w)$ is supermodular over D/ρ, where $+, -, *, /$ are taken entry-wise.*

Proof. The results follow from the definition. □

We note that when $D = Q$, d is in Q. We observe a useful property of g.

Proposition 5.7. *s is supermodular over Q and $\mathrm{convenv}_{\mathcal{U}}(s) = \mathrm{convenv}_{\mathcal{Q}}(g)$.*

Proof. According to Example 2.6.2 of [287], the signomial term ψ_α with $\alpha > 0$ is a Cobb-Douglas function, which is supermodular over $\mathbb{R}^h_+$. This implies that the power function ψ_β is supermodular over $\mathbb{R}^h_+$. By Lemma 5.6, s is supermodular over $\mathcal{U} = [0, 1]^h$. As $Q = \{0, 1\}^h$ is a product subset of $\mathcal{U}$, s is supermodular over Q. After the scaling and translation, s is still concave, so it follows from Lemma 3.5 that $\mathrm{convenv}_{\mathcal{U}}(s) = \mathrm{convenv}_{\mathcal{Q}}(s)$. □

The search for facets of s can be reduced to a more general problem, which involves finding facets of supermodular functions over Boolean hypercubes.

We note that both power functions and multilinear terms can be considered Cobb-Douglas functions. Consequently, a similar argument can be used to demonstrate that multilinear terms are supermodular over any product subset of $\mathbb{R}^h_+$.

5.3.2 Convex envelopes of bivariate supermodular functions

Using the aforementioned result in Sect. 4.2, we can construct an envelope-inducing family for bivariate supermodular functions. Let

$$S^2_1 \coloneqq \{00, 10, 01\}, S^2_2 \coloneqq \{11, 10, 01\}. \tag{5.10}$$

One can find that $\mathrm{conv}(S^2_1) = \{(w_1, w_2) \in [0, 1]^2 : w_1+w_2 \leq 1\}, \mathrm{conv}(S^2_2) = \{(w_1, w_2) \in [0, 1]^2 : w_1 + w_2 \geq 1\}$ are two triangles in $[0, 1]^2$. We have that

$$\begin{aligned} f_{S^2_1}(w) &= f(00) + (f(10) - f(00))w_1 + (f(01) - f(00))w_2, \\ f_{S^2_2}(w) &= f(11) + (f(01) - f(11))(1 - w_1) + (f(10) - f(11))(1 - w_2). \end{aligned}$$

We show that these two affine functions define the convex envelope of f.

Theorem 5.8. *For $h = 2$, $\{S^2_k\}_{k\in[2]}$ as in* (5.10) *is envelope-inducing family in Q.*

Proof. It is easy to see that for all $k \in [2]$, S^2_k is affinely independent and $\{\mathrm{conv}(S^2_k)\}_{k\in[2]}$ is a triangulation of $\mathcal{U}$. Therefore, it suffices to show that $\{S^2_k\}_{k\in[2]}$ is facet-inducing, i.e., $f_{S^2_1}, f_{S^2_2}$ are affine underestimators of f.

Case i. We note that for all $w \in S^2_1 = \{00, 10, 01\}$, $f_{S^2_1}(w) = f(w)$. Note that $Q \setminus S^2_1 = \{11\}$. It follows from the definition of the affine function $f_{S^2_1}$ that

$$f_{S^2_1}(11) = f_{S^2_1}(10) + (f_{S^2_1}(01) - f_{S^2_1}(00)) = f(10) + (f(01) - f(00)).$$

It follows from the supermodularity of f that

$$f(10) + (f(01) - f(00)) \leq f(10) + (f(11) - f(10)) = f(11).$$

Thereby, $f_{S_1^2}$ underestimates f.

Case ii. We note that for all $w \in S_2^2 = \{11, 10, 01\}$, $f_{S_2^2}(w) = f(w)$. Note that $Q \setminus S_2^2 = \{00\}$. It follows from the definition of the affine function $f_{S_2^2}$ that

$$f_{S_2^2}(00) = f_{S_2^2}(10) - (f_{S_1^2}(11) - f_{S_1^2}(01)) = f(10) + (f(11) - f(01)).$$

It follows from the supermodularity of f that

$$f(10) - (f(11) - f(01)) \leq f(10) - (f(10) - f(00)) = f(00),$$

which concludes the proof. □

5.3.3 Convexity and reverse-convexity

Our cutting algorithm can detect the convexity/reverse-convexity of signomial term sets. The detection is simply through normalized DDC formulations.

Denote by e_j^ℓ and e_j^h the j-th unit vector in $\mathbb{R}^h$ and $\mathbb{R}^\ell$ respectively. Then, we have the following observations:

i) if $\|\beta\|_1 = 1, \gamma = 0$, i.e., ψ_β is concave and ψ_γ is 1, then $\mathcal{S}_{\text{st}}$ is reverse-convex;

ii) if $\|\beta\|_1 \leq 1, \gamma = e_j^\ell$ for some $j \in [\ell]$, i.e., ψ_β is concave and ψ_γ is a linear univariate function, then $\mathcal{S}_{\text{st}}$ is reverse-convex;

iii) if $\beta = e_j^h, \|\gamma\|_1 \leq 1$ for some $j \in [h]$, i.e., ψ_β is a linear univariate function and ψ_γ is concave, then $\mathcal{S}_{\text{st}}$ is convex;

iv) if $\|\beta\|_1 = 0, \|\gamma\|_1 = 1$, i.e., ψ_β is 1 and ψ_γ is concave, then $\mathcal{S}_{\text{st}}$ is convex.

We note that similar results are found in [86, 220]. The results in [86] are proved by checking the negative/positive-semidefiniteness of the Hessian matrix of a signomial term. According to the normalized DCC formulation, the results are evident.

5.4 Computational results

In this section, we conduct computational experiments to assess the efficiency of the proposed valid inequalities.

The MINLPLib dataset comprises instances of MINLP problems that involve signomial terms, and some of these instances are SP problems. To build our benchmark, we select instances from MINLPLib that satisfy the following criteria: (i) the instance includes signomial functions or polynomial functions, (ii) the continuous relaxation of the instance is non-convex. Our benchmark consists of a diverse set of 251 instances in which nonlinear functions consist of signomial and other functions. These problems frequently arise in practical applications and are commonly solved by general-purpose solvers.

Experiments are conducted on a server with Intel Xeon W-2245 CPU @ 3.90GHz, 126GB main memory, and Ubuntu 18.04 system. We use SCIP 8.0.3 [62] as the framework for reading and solving problems, as well as conducting cut separation. SCIP is integrated with `CPLEX` 22.1 as LP solver and `IPOPT` 3.14.7 as NLP solver.

We evaluate the efficiency of the proposed valid inequalities in four different settings. The first setting, denoted as `disable`, does not apply any of the proposed valid inequalities. The second setting, denoted as `oc`, applies only the outer approximation cuts. The third setting, denoted as `ic`, applies only the intersection cuts. The fourth setting combines both the `oc` and `ic` settings by applying both cuts. We let SCIP's default internal cuts to handle univariate signomial terms and multilinear terms. Our valid inequalities only handle the other high-order signomial terms. The source code, data, and detailed results can be found in our online repository: github.com/lidingxu/ESPCuts.

In our benchmark, there are 150 instances classified as *affected*, in which at least one of the `oc`, `ic`, and `oic` settings adds cuts. There are 86 instances among the affected ones, for which SCIP's default configuration (i.e., the `disable` setting) runs at least 500 seconds. Such instances are classified as *affected-hard*. Each test run uses SCIP configured by a setting to solve an instance. To solve the instances, we use the SCIP solver with its sBB algorithm, imposing a time limit of 3600 seconds. For each test run, we measure the running time, the number of sBB search nodes, and the relative open duality gap.

To aggregate the performance metrics for a given setting, we compute shifted geometric means (SGMs) over our test set. The SGM for the running time incorporates a shift of 1 second. The SGM for the node number incorporates a shift of 100 nodes. The SGM for the relative gap incorporates a shift of 1%. We also compute SGMs of performance metrics over the subset of affected and affected-hard instances. The performance results are presented in Table 5.1, where we also compute the relative values of SGMs of performance metrics compared to the `disable` setting. Our following analysis is based on the results on affected and affected-hard instances.

Setting		All				Affected				Affected-hard			
		solved	nodes	time	gap	solved	nodes	time	gap	solved	nodes	time	gap
`disable`	absolute	138/251	6510.5	122.0	4.7%	71/150	15592.4	253.6	5.7%	7/86	175973.8	3600.0	26.7%
	relative		1.0	1.0	1.0		1.0	1.0	1.0		1.0	1.0	1.0
`oc`	absolute	140/251	5954.1	118.0	4.5%	73/150	13443.9	241.4	5.4%	10/86	115262.3	2872.7	23.3%
	relative		0.91	0.97	0.97		0.86	0.95	0.95		0.65	0.8	0.87
`ic`	absolute	140/251	6144.3	122.4	4.4%	73/150	14081.5	252.1	5.2%	10/86	128072.7	2994.1	22.0%
	relative		0.94	1.0	0.95		0.9	0.99	0.91		0.73	0.83	0.82
`oic`	absolute	139/251	5934.6	117.7	4.6%	72/150	13275.6	236.8	5.6%	10/86	118054.1	2758.3	23.0%
	relative		0.91	0.96	0.99		0.85	0.93	0.98		0.67	0.77	0.86

Table 5.1 Summary of performance metrics on MINLPLib instances

First, we observe that the proposed valid inequalities lead to the successful solution of 2 additional instances compared to the `disable` setting. Considering the `oc` setting, it solves 2 more instances than the `disable` setting. Note that, in the affected-hard benchmark, the number of solved instances by any non-disable setting is 3 more than that by the disable setting; in the affected benchmark, the number of solved instances by any non-disable setting is at most 2 more than that by the disable setting. This is because restriction of the benchmark can reduces more solvable instances by the disable setting.

The reductions in the running time, and relative gap achieved by the `oc` setting are respectively 5%, 5% for affected instances and 20%, 13% for affected-hard instances. For the `ic` setting, it solves 2 more instances than the `disable` setting. The reductions in the running time, and relative gap achieved by the `ic` setting are respectively 1%, 9% for affected instances and 17%, 14% for affected-hard instances. In the case of the `oic` setting, it solves 1 additional instance compared to the `disable` setting. The reductions in the running time, and relative gap achieved by the `oic` setting are respectivel 7%, 2% for affected instances and 23%, 14% for affected-hard instances.

We note that the running time does not give much information on affected-hard instances, because only 10 instances can be solved within 3600 seconds. For these instances, the gap reduction is more useful to measure the reduction of the search space by the proposed valid inequalities. However, for all affected instances, the running time is still important, as it measures the acceleration by the valid inequalities.

Secondly, we observe that all cut settings have a positive impact on the performance of SCIP, although the extent of reduction varies. When comparing the `oc` and `ic` settings, we find that the `oc` setting leads to a greater reduction in running time. This difference in running time arises because computing intersection cuts involves extracting a simplicial cone from the LP relaxation and applying bisection search along each ray of the cone. These procedures require more computational resources compared to the construction of outer approximation cuts.

On the other hand, the `ic` setting demonstrates better performance in terms of gap reduction. Intersection cuts approximate the intersection of a signomial term set with the simplicial cone, while outer approximation cuts approximate the intersection of a signomial term set with a hypercube. Around the relaxation point, the simplicial cone typically provides a better approximation than the hypercube. Hence, `ic` achieves a larger reduction in the relative gap. However, the better simplicial conic approximation does yield a significant improvement compared to the hypercubic approximation.

Lastly, the `oic` setting combines both the `oc` and `ic` settings, achieving the best reduction in the running time. However, for affected and affected-hard instances, the setting exhibits different results on gap reduction. In fact, the results of affected-hard instances give more insights, since the goal of valid inequalities is to accelerate the convergence for had instances. In this sense, the `oic` setting achieves nearly the best result, so it inherits the best of both valid inequalities. However, its improvement compared to the individual settings is not significant.

To summarize, the performances of the `oc` and `ic` settings are comparable. They can lead to smaller duality gap with less computational time, which are desirable for solvers, and one can use any of them. Moreover, they do not hurt each other.

5.5 Conclusion

In this chapter, we study valid inequalities for SP problems, and propose two types of valid linear inequalities: intersection cuts and outer approximation cuts. They are both derived from the normalized DCC formulations of signomial term sets. First, we study general conditions on maximal $\mathcal{S}$-free sets. We construct maximal signomial-term-free sets, from which we generate intersection cuts. Secondly, we construct convex outer approximations of signomial term sets within hypercubes. We provide extended formulations for the convex envelopes of the

concave functions in the normalized DCC formulations. Then we separate valid inequalities for the convex outer approximations through projection. Additionally, when $h = 2$, we use supermodularity to derive a closed-form expression for the convex envelopes.

We present a comparative analysis of computational results obtained from the MINLPLib instances. This analysis demonstrates the effectiveness of the proposed valid inequalities. The results indicate that intersection cuts and outer approximation cuts exhibit similar performance, and their combination inherits the best of the individual settings. In particular, it is straightforward to implement outer approximation cuts in general-purpose solvers. In the future, we intend to carefully fine-tune outer approximation cuts and develop it as an easy-to-use plugin.

We currently deal with signomial terms that explicitly present in the signomial lift, but our results can be extended to handle more "faces" of the signomial lift. In the future, the proposed valid inequalities can approximate nonlinear aggregations of constraints defining the signomial lift. Specifically, given signomial constraints $\{\psi_{\alpha^i}(x) = y_i\}_{i\in[r]}$, with any exponent vector $\zeta \in \mathbb{R}^r$, we can employ *signomial aggregation* to generate a new signomial constraint: $\psi_{(\sum_{i\in[r]} \zeta_i \alpha^i)}(x) = \psi_\zeta(y)$. This constraint is valid for the signomial lift and encodes more variables and terms. Subsequently, we can apply DCC reformulation to the constraints $\psi_{(\sum_{i\in[r]} \zeta_i \alpha^i)}(x) \leq \psi_\zeta(y)$ and $\psi_{(\sum_{i\in[r]} \zeta_i \alpha^i)}(x) \geq \psi_\zeta(y)$. Finally, we can separate the proposed valid inequalities. As far as we know, the signomial aggregation operator is not used yet for polynomial programming, as it outputs a signomial constraint.

Chapter 6

Intersection cuts for submodular optimization

6.1 Introduction

In this chapter, we first consider $\mathcal{S}$ as the Boolean-hypograph $\mathrm{hypo}_{\{0,1\}^n}(f)$ of f. We use convex extensions of f in order to construct some $\mathcal{S}$-free sets, which we call *Boolean-hypograph-free*. The Boolean-hypograph set $\mathrm{hypo}_{\{0,1\}^n}(f)$ is a specialization of the constraint set $\{(x,t) \in \{0,1\}^n \times \mathbb{R} : f_1(x) - f_2(x) \geq \ell t\}$ with $\ell \in \{0,1\}$. Then, we consider $\mathcal{S}$ as this general constraint set and extend our results to handle this general case. Finally, we propose an efficient algorithm to compute intersection cuts derived from $\mathcal{S}$-free sets. To the best of our knowledge, intersection cuts have not been applied directly to approximate problems with submodular and/or supermodular structures.

We implement intersection cuts within the SCIP solver [61] and test them on MAX CUT, PSEUDO BOOLEAN MAXIMIZATION, and BAYESIAN D-OPTIMAL DESIGN problems. We show the strengths and weaknesses of intersection cuts under these different settings.

6.1.1 Literature review

The *base inequalities* [237] are a class of valid linear inequalities for the hypographs of general submodular functions. For a class of special submodular functions, lifting procedures [8, 277] can strengthen the base inequalities. The base inequalities can be separated either using heuristics [8] or a Benders-like framework [104] if the point to be separated is integer. The method defined in [23] combines valid inequalities for the submodular and supermodular components of an SS function. We refer to [19, 20, 65, 72, 171, 189, 260, 274, 312, 316] for more details about the exploitation of submodular/supermodular functions in mathematical programs. Supermodular polynomials in binary variables are defined and studied in [65, 260]. The submodularity of the D-OPTIMAL DESIGN problem is exploited in [266, 274].

As already mentioned, intersection cuts generate valid inequalities for sets that are hard to optimize over. Gomory introduced the corner polyhedron [159], and his celebrated mixed-integer cuts [160] are special intersection cuts derived from split disjunctions [236]. The definition of intersection cuts for arbitrary set $\mathcal{S}$ is due to [125, 156]. We refer to [16, 17, 39, 40, 101, 102, 109, 119, 125, 261] for a more in-depth analysis. The method given

in [289] can generate valid inequalities that cut off points outside $\mathcal{S}$-free sets. We refer to [17, 46, 194, 195, 229, 230] for relevant recent developments in mixed-integer conic programming.

For the cases where the nonconvexity of $\mathcal{S}$ is not just due to integer variables, we refer to [137] for bilevel programs, [64] for outer-product sets, [233, 232] for quadratic constraint sets, [312] for signomial-term sets, and [138] for bilinear sets. The method given in [271] constructs intersection cuts for sets arising from factorable programs that contain DC functions [188].

Next, we discuss valid inequalities for polynomial programming, because we use polynomial programs in binary variables as a benchmark in our computational study. In [64], intersection cuts approximate a nonconvex lifted set, namely the outer product set arising from the extended formulation of a polynomial program. Lifted sets link decision variables to auxiliary variables representing (graphs of) monomials up to a given degree. We remark that in most combinatorial optimization problems, decision variables are binaries. The polynomial program of interest is then a Boolean Multilinear Program (BMP). The corresponding lifted set is the *Boolean multilinear set* [111, 139], the convex hull of which is the so-called Boolean multilinear polytope. Valid inequalities for the Boolean multilinear polytope may be stronger than those for the convex hull of the outer product set. Various Gomory-Chvátal-based inequalities [115–118] are valid for the multilinear polytope. The separation and strength of these inequalities depend on the hypergraph representing the underlying sparsity pattern of the multilinear set.

We consider a constrained polynomial program, and assume that some of its constraints are neither integrality constraints nor variable bound constraints. After lifting, those constraints are linear and thus define a convex set $\mathcal{S}_1$. The lifted set $\mathcal{S}_2$ is nonconvex, and $\mathcal{S}_1 \not\subseteq \mathcal{S}_2$. The polynomial program is then equivalent to linear optimization over $\mathrm{conv}(\mathcal{S}_1 \cap \mathcal{S}_2)$. However, in general, $\mathrm{conv}(\mathcal{S}_1 \cap \mathcal{S}_2) \neq \mathcal{S}_1 \cap \mathrm{conv}(\mathcal{S}_2)$, so the convexification of the lifted set may not yield an equivalent convex problem. To address this issue, one attempt is to directly consider $\mathrm{conv}(\mathcal{S}_1 \cap \mathcal{S}_2)$ and generate valid inequalities for it. Some work in this sense exists for certain interesting special cases, e.g. the intersection of multilinear sets with additional constraint sets such as cardinality constraints [87]. Another attempt is to consider constraints in projected formulations, e.g., in mixed-integer quadratically constrained quadratic programs [268]. Since the representation complexity of the projected formulation is smaller than that of the extended formulation, this approach is also amenable to computation. In [89, 233], intersection cuts for the set defined by a quadratic constraint are derived. If additionally, some of the nonbasic variables of the LP relaxation need to be integer, the monoidal technique [90] can strengthen such intersection cuts.

However, generating valid inequalities for Boolean multilinear constraints, and, more generally, constructing $\mathcal{S}$-free sets for nonlinear constraints on discrete variables, remain problems of considerable interest. In this chapter, we look at these questions through a "submodularity lens".

6.1.2 Contribution

In Sect. 4.1.3, we already studied vairous properties for $\mathcal{S}$-free sets arising in submodular maximization. We summarize those theoretical contributions here. Our primary contribution is the construction of Boolean-hypograph-free sets. We show that a maximal Boolean-hypograph-free set $\mathcal{C} \times \mathbb{R}$ can be lifted from a maximal $\{0,1\}^n$-free set. We also give an alternative construction of Boolean-hypograph-free sets by exploiting the submodularity. We relate the

analytical properties of $\bar{\mathsf{F}}_f$ in (4.10) to its combinatorial properties, which inherit those of the Lovász extension. We show that the epigraph $\mathrm{epi}(\bar{\mathsf{F}}_f)$ of $\bar{\mathsf{F}}_f$ is a Boolean-hypograph-free set that is larger than the epigraph of the Lovász extension. However, unlike in the continuous setting, $\mathrm{epi}(\bar{\mathsf{F}}_f)$ is not maximally Boolean-hypograph-free. We give necessary and sufficient conditions on maximal Boolean-hypograph-free sets that contain $\mathrm{epi}(\bar{\mathsf{F}}_f)$.

The second contribution is the computation of intersection cuts. We reduce the intersection cut separation problem to solving univariate nonlinear equations, which we achieve by a hybrid discrete Newton algorithm like [157]. We show that facets of $\mathrm{epi}(\bar{\mathsf{F}}_f)$ can be separated in strongly polynomial time. This implies that the (sub)-gradients required by the Newton algorithm can be computed in a strongly polynomial time. The hybrid discrete Newton algorithm finds a zero point of a univariate nonlinear equation in a finite number of steps. By contrast, the conventional bisection algorithm only guarantees ϵ-approximated solutions for $\epsilon > 0$.

Lastly, we extend the previous findings to constraint sets involving an SS function. We show that any Boolean multilinear function is an SS function. This result yields intersection cuts for multilinear constraints in binary variables.

6.1.3 Outline of the chapter

The rest of the chapter is organized as follows. In Sect. 6.2, we consider applications for intersection cuts to Boolean multilinear constraints and BAYESIAN D-OPTIMAL DESIGN. In Sect. 6.3, we propose the hybrid discrete Newton algorithm for computing intersection cuts. In Sect. 6.4, we analyze the computational results.

6.2 Application

In this section, we discuss the application of intersection cuts to Boolean multilinear programming and D-optimal design. We exploit the submodular structures in these two problems.

6.2.1 Boolean multilinear constraints

We consider the construction of $\mathcal{S}$-free sets for Boolean multilinear constraints. Since $x \in \{0,1\} \Leftrightarrow x^2 = x$, one can reduce a polynomial function defined on binary variables to a multilinear function, whose monomials do not include powers. For example, $x_1^2 x_3^3 + x_2^2$ can be reduced to $x_1 x_3 + x_2$. A Boolean multilinear function is sometimes called a pseudo Boolean function.

A similar case is the construction of $\mathcal{S}$-free sets for continuous quadratic constraints [233]. We call this construction the "continuous approach". It applies eigenvalue decomposition to factor the symmetric matrix representing quadratic terms in a quadratic constraint. Through this factorization, the quadratic constraint is reformulated to a DC constraint, possibly intersected with additional linear constraints. This reformulation is amenable to the reverse-linearization technique. Applying the technique with possibly additional operations, one can construct the so-called continuous-quadratic-free sets [233] [1]. Multilinear terms, however, are represented by tensors. High-order tensor decomposition is more complicated than matrix decomposition

[1] The construction is *de facto* discussed case by case. For some cases, the reverse-linearization technique already suffices to produce continuous-quadratic-free sets. For other cases, one needs additional operations, e.g., projecting out a lineality space. Notably, all cases require the eigenvalue decomposition and its resulting DC constraint.

[193]. It is doubtful whether the continuous approach can be extended so as to produce DC functions from tensors.

Here we consider an alternative discrete approach. It exploits the submodularity and the supermodularity of Boolean multilinear functions. In [65, 237], a class of Boolean multilinear functions is shown to be supermodular. We give a submodular-supermodular decomposition for general Boolean multilinear functions in the following.

Proposition 6.1. *Consider a Boolean multilinear function $f : \mathcal{B} \to \mathbb{R}, x \mapsto \sum_{k\in[K]} a_k \prod_{j\in A_k} x_j$ with K multilinear terms, where $A_k \subseteq [n]$. Let $f = f_1 - f_2$ where*

$$f_1(x) := \sum_{\substack{k\in[K]\\ a_k<0}} a_k \prod_{j\in A_k} x_j \tag{6.1}$$

$$f_2(x) := -\sum_{\substack{k\in[K]\\ a_k>0}} a_k \prod_{j\in A_k} x_j. \tag{6.2}$$

Then f_1, f_2 are submodular over $\mathcal{B}$.

Proof. It follows from Theorem 13.21 of [112] that f_1, f_2 are submodular functions over $\mathcal{B}$. □

Since every Boolean multilinear function is an SS function, we can construct $\mathcal{S}$-free sets for the corresponding Boolean-superlevel set or Boolean-hypograph set.

Corollary 6.2. *Consider a multilinear function $f : \mathcal{B} \to \mathbb{R}$, where $f(x) = \sum_{k\in[K]} a_i \prod_{j\in A_k} x_j$ for $A_k \subseteq [n]$ as in Prop. 6.1, and $f_1(x), f_2(x)$ as in Eq.* (6.1)-(6.2)*. Let $\mathcal{S}$, $\overline{\mathcal{S}}$, and $\mathcal{C}_{\tilde{x}}$ be as* (4.14)*,* (4.15)*,* (4.16)*, respectively. Then, the set $\mathcal{C}_{\tilde{x}}$ is an $\mathcal{S}$-free set. Moreover, if $\tilde{x} \notin \overline{\mathcal{S}}$, then $\mathcal{C}_{\tilde{x}}$ does not contain $\tilde{x}$ in its interior.*

Proof. By Prop. 6.1, we know that both f_1 and f_2 are submodular. Hence, the result follows by applying Prop. 4.27. □

Importing the notation in Prop. 6.1, a BMP problem has the following form:

$$\max \quad t \tag{6.3a}$$

$$\sum_{k\in\mathcal{K}_0} a_{ik} \prod_{j\in A_k} x_j \geq t \tag{6.3b}$$

$$\forall i \in [m] \quad \sum_{k\in\mathcal{K}_i} a_{ik} \prod_{j\in A_k} x_j \geq 0 \tag{6.3c}$$

$$\forall j \in [n] \quad x_j \in \{0,1\}, \tag{6.3d}$$

where m is the number of constraints, K is the number of distinct multilinear terms in the BMP, $\mathcal{K}_i \subseteq [K]$ is the index set of multilinear terms in the i-th constraint (0 for objective). Unconstrained BMP has several synonyms: PSEUDO BOOLEAN MAXIMIZATION or MULTILINEAR UNCONSTRAINED BINARY OPTIMIZATION (MUBO).

To construct $\mathcal{S}$-free sets for Boolean multilinear constraints in the BMP, we need to write them as the standard form (4.14). For all $i \in [m]$ or $i = 0$, let

$$f_i(x) := \sum_{k\in\mathcal{K}_i} a_{ik} \prod_{j\in A_k} x_j,$$

and write

$$f_i(x) = f_{i1}(x) - f_{i2}(x),$$

where $f_{i1} := \sum_{k\in\mathcal{K}_i:a_{ik}<0} a_{ik} \prod_{j\in A_k} x_j$ and $f_{i2} := -\sum_{k\in\mathcal{K}_i:a_{ik}>0} a_{ik} \prod_{j\in A_k} x_j$ are two submodular functions.

The objective and constraints of (6.3) can be represented as

$$f_{i1}(x) - f_{i2}(x) \geq \ell_i t$$

(for all $i \in [m]$, $\ell_i = 0$, and ℓ_0 = 1), which, by Cor. 6.2, is in the standard form.

Separating intersection cuts requires LP relaxations or simplicial cones. One can first lift multilinear terms to obtain an extended formulation:

$$\max \quad t \tag{6.4a}$$

$$\sum_{k\in\mathcal{K}_0} a_{0k} y_k \geq t \tag{6.4b}$$

$$\forall i \in [m] \quad \sum_{k\in\mathcal{K}_i} a_{ik} y_k \geq 0 \tag{6.4c}$$

$$k \in [K] \qquad y_k = \prod_{j\in A_k} x_j \tag{6.4d}$$

$$\forall j \in [n] \qquad x_j \in \{0,1\} \tag{6.4e}$$

The standard Boolean linearization technique [111] can reformulate a multilinear term $\prod_{j\in A_k} x_j$ by its underestimators and overestimators:

$$\forall j \in A_k \quad y_k \leq x_j \tag{6.5a}$$

$$y_k \geq |A_k| + 1 - \sum_{j\in A_k} x_j, \tag{6.5b}$$

where $|A_k|$ is the cardinality of A_k. Then, by linearizing each nonlinear constraint (6.4d) as linear constraints in (6.5), one obtains a MILP reformulation of (6.4).

To construct LP relaxations, one can simply drop the integrality constraints $x_j \in \{0,1\}$. The direct LP relaxation of the MILP reformulation is also an LP relaxation of the BMP (6.4). Following the method at the end of Sect. 4.1.3, we can construct an optimal tableau cone in the extended space (x, y, t). The $\mathcal{S}$-free set belongs to a projected space (i.e., (x, t)-space). By extracting the (x, t) entries of the rays of the optimal tableau cone, we project the optimal tableau cone into the (x, t)-space. Given the projection of this optimal tableau cone, it is straightforward to construct intersection cuts for the BMP: we separate the intersection cuts constructed by means of the $\mathcal{S}$-free sets given by Prop. 4.27.

As explained above, Boolean quadratic constraints belong to Boolean multilinear constraints, and continuous quadratic constraints relax Boolean quadratic constraints. Both the continuous and discrete approaches can construct valid $\mathcal{S}$-free sets for Boolean quadratic constraints. We remark that maximal continuous-quadratic-free sets are no longer maximally Boolean-quadratic-free. It is easy to see that the discrete approach preserves the term-wise sparsity patterns of the SS functions and requires no factorizations. Therefore, the discrete approach is computationally amenable to ill-conditioned or sparse coefficient matrices.

6.2.2 D-optimal design

In statistical estimation, optimal designs are a class of experimental designs that are optimal with respect to some statistical criterion. We derive an extended convex MINLP formulation for the BAYESIAN D-OPTIMAL DESIGN problem. In this formulation, the problem is a cardinality-constrained submodular maximization problem.

Let $\mathbb{S}^m$ denote the set of m-by-m symmetric matrices, and let $\mathbb{S}^m_+$ (resp. $\mathbb{S}^m_{++}$) denote the set of m-by-m positive semi-definite (resp. positive definite) matrices. Given a set of full row-rank matrices $\{M_j \in \mathbb{R}^{m\times r_k}\}_{j\in[n]}$, an optimal design problem usually has the following form:

$$\max \quad \Phi(\sum_{j\in[n]} M_j {M_j}^\top x_j) \tag{6.6a}$$

$$\sum_{j\in[n]} x_j = k \tag{6.6b}$$

$$\forall j \in [n] \qquad x_j \in \{0,1\}, \tag{6.6c}$$

where k is the size of the design and $\Phi : \mathbb{S}^m \to \mathbb{R}$ is the design criterion. The matrix $M(x) := \sum_{j\in[n]} M_j {M_j}^\top x_j$ is called the *information matrix*. For the D-optimal criterion [72, 267], Ψ is the log determinant function $\log\det$.

Reseachers usually study BAYESIAN D-OPTIMAL DESIGN, where a statistical prior on the data $\{M_i\}_{i\in[n]}$ adds a regularization term ϵI into the information matrix $M(x)$. Thus, $M(x) = \epsilon I + \sum_{j\in[n]} M_j {M_j}^\top x_j$. The additional term is also due to the well-posedness: when $x = 0$, we have that $\log\det(M(0)) = \log\det(\epsilon I)$ is well defined. Then, the submodular maximization version of the BAYESIAN D-OPTIMAL DESIGN problem has the following formulation:

$$\max \quad \log\det\left(\epsilon I + \sum_{j\in[n]} M_j {M_j}^\top x_j\right) \tag{6.7a}$$

$$\sum_{j\in[n]} x_j = k \tag{6.7b}$$

$$\forall j \in [n] \qquad x_j \in \{0,1\}, \tag{6.7c}$$

The log determinant function is concave and has a semi-definite programming (SDP) and geometric programming representation [18]. The scalability of the mixed-integer log determinant formulation above is limited by the current state of SDP solvers. Based on the second order cone representation of the determinant function $\det(M(x))$ [267], we give an

extended formulation for (6.7):

$$\max \quad t \tag{6.8a}$$

$$t \leq \sum_{i\in[m]} \log(J_{ii}) \tag{6.8b}$$

$$\sum_{j\in[n]\cup\{0\}} M_j Z_j = J \tag{6.8c}$$

$$J \text{ is lower triangular} \tag{6.8d}$$

$$j \in [n]\cup\{0\}\, i \in [m] \quad \|Z_j e_i\|^2 \leq u_{ji} x_j \tag{6.8e}$$

$$i \in [m] \quad \sum_{j\in[n]\cup\{0\}} u_{ji} \leq J_{ii} \tag{6.8f}$$

$$\sum_{j\in[n]} x_j = k \tag{6.8g}$$

$$x \in \{1\} \times \mathcal{B} \tag{6.8h}$$

$$J \in \mathbb{R}^{m\times m} \tag{6.8i}$$

$$j \in [n]\cup\{0\} \quad Z_j \in \mathbb{R}^{r_j\times m} \tag{6.8j}$$

$$j \in [n]\cup\{0\}\, i \in [m] \quad u_{ji} \in \mathbb{R}_+^{r_j\times m}, \tag{6.8k}$$

where $M_0 = \epsilon^{1/2} I$ is an auxiliary matrix. One can represent this formulation by low-dimensional convex cones [18], e.g., (rotated) second-order cones, and exponential cones. Therefore, this extended formulation is amenable to computation.

Proposition 6.3. (6.8) *is equivalent to* (6.7)*, and the objective function of* (6.7) *is submodular w.r.t.* x*.*

Proof. One can modify the original D-optimal design problem by adding a slack variable $x_0 = 1$. Applying the logarithmic transformation to results in [267], (6.8) is equivalent to (6.7). It follows from [266, 274] that (6.7) is submodular w.r.t. x. □

A global optimization solver like SCIP can linearize the constraints in the extended formulation (6.8), and thus produces an LP relaxation in the extended space. We can obtain an optimal tableau cone as the approach dealing with the BMP. Then, we can construct intersection cuts from Boolean-hypograph-free sets.

6.3 Separation problem

In this section, we consider the separation problem for an intersection cut from an $\mathcal{S}$-free set. Summarizing the previous sections, the $\mathcal{S}$-free set is in the form of

$$\mathcal{C} := \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \mathsf{G}(x) \leq \ell t\},$$

where $\mathsf{G}(x) = \max_{s\in\mathrm{ext}(\mathsf{EPM}_g)} sx$ is the extended envelope of some submodular function g over $\mathcal{B}$ and $\ell \in \{0,1\}$. We remark that the extended envelope epigraph EE_f in (4.8) is a special case with $\ell = 1$ and $g = f$; the set $\mathcal{C}_{\tilde{x}}$ in (4.16) is also a special case that $g(x) = f_1(x) - \gamma^* x$.

Assume that $z^* := (\tilde{x}, \tilde{t})$ is the vertex of an optimal tableau cone $\mathcal{R}$, and $z^* \in \mathrm{int}(\mathcal{C})$. Recalling the cut coefficient formula in Sect. 3.4, the separation problem consists in computing

the step length along each ray r^j:

$$\eta_j^* = \sup_{\eta_j \geq 0} \{\eta_j : z^* + \eta_j r^j \in \mathcal{C}\}. \tag{6.9}$$

This line search problem asks for the step length to the border of $\mathcal{C}$ along the ray r^j from z^* which, we recall, is an interior point of $\mathcal{C}$. We denote by r_x^j, r_t^j the projection of r^j on x- and t-spaces. Looking at the function defining $\mathcal{C}$, the intersection step length η_j^* is the zero point of the following function:

$$\zeta^j : \mathbb{R}_+ \to \mathbb{R}, \text{ where } \zeta^j(\eta_j) = \ell(\tilde{t} + r_t^j \eta_j) - \mathsf{G}(\tilde{x} + r_x^j \eta_j).$$

This function enjoys the following properties.

Proposition 6.4. *ζ^j is a concave piece-wise linear function over $[0, +\infty]$ with $\zeta^j(0) > 0$. If $\eta_j^* < \infty$ and there exists an $\eta_j' > 0$ with $\zeta^j(\eta_j') = 0$, then $\eta_j' = \eta_j^*$, i.e., the solution η_j^* must be unique. For all $s^* \in \operatorname{argmax}_{s \in \operatorname{ext}(\mathsf{EPM}_g)} s(\tilde{x} + \eta_j r_x^j)$, $\ell r_t^j - s^* r_x^j$ is a subgradient in $\partial\zeta^j(\eta_j)$. For $\eta_j > \eta_j^*$, $\partial\zeta^j(\eta_j) \leq \partial\zeta^j(\eta_j^*)$.*

Proof. Since the extended envelope G is the maximum of linear functions, it is convex and piece-wise linear, so ζ^j is concave and piece-wise linear. Since $\zeta^j(0) = \ell\tilde{t} - \mathsf{G}(\tilde{x})$, it follows from the assumption $z^* \in \operatorname{int}(\mathcal{C})$ that $\ell\tilde{t} > \mathsf{G}(\tilde{x})$ and thus $\zeta^j(0) > 0$. Since $\mathcal{C}$ is closed and convex, $\eta_j' = \eta_j^*$ if and only if $z^* + \eta_j' r^j \in \operatorname{bd}(\mathcal{C})$. That is $\mathsf{G}(r_x^j \eta_j + \tilde{x}) = \mathsf{G}(\tilde{x}) + r_t^j \eta_j'$, i.e., $\zeta^j(\eta_j') = 0$. Since $s^* \in \partial\mathsf{G}(\tilde{x} + r_x^j \eta_j)$, by the chain rule, $\ell r_t^j - s^* r_x^j$ is a subgradient of ζ^j. By the concavity of ζ^j, its subgradients are non-increasing. □

By Prop. 6.4, the line search problem (6.9) is reduced to solving the univariate nonlinear equation:

$$\zeta^j(\eta^j) = 0. \tag{6.10}$$

For each ray r^j, solving (6.10) gives the unique zero point of the univariate function ζ^j, or certifies that no such point exists.

To solve the univariate nonlinear equation (6.10), it is natural to deploy a Newton-like algorithm. Therefore, we need the value and (sub)gradient information of ζ^j: the computation of ζ^j can then be reduced to the computation of G. The value and subgradients of G are obtained by means of a sorting algorithm (see Prop. 4.14). We note that these computations can be carried out in strongly polynomial time.

Previous works [90, 312] use the bisection algorithm, which guarantees finding the zero point within a given tolerance. Our implementation, which we call *hybrid discrete Newton algorithm*, is a combination of the discrete Newton algorithm [157] and the bisection algorithm. The role of the bisection algorithm in Alg. 6.1 is to help find a starting point for the Newton algorithm. Thanks to the piece-wise linearity of the univariate function ζ^j, our algorithm finds an exact zero point in a finite time.

Proposition 6.5. *The hybrid discrete Newton algorithm terminates in a finite number of steps and finds the zero point η_j^*.*

Proof. For all $\eta \in \mathbb{R}_+$, we assume that Algorithm 6.1 chooses and computes a unique subgradient β at η_j, we denote it $\nabla\zeta^j(\eta_j)$, and call it algorithmic gradient. The concavity of ζ^j implies that its algorithmic gradient is monotone-decreasing w.r.t. η_j. There is a threshold

Algorithm 6.1: Hybrid discrete Newton algorithm

Input: The univariate function ζ^j, (scalar) starting point $\Delta > 0$ (default: 0.2), a numeric η_∞ representing $+\infty$, and the maximum number I of search steps (default: 500);
Output: $\eta_j > 0$ such that $\zeta^j(\eta_j) = 0$;
Let step number $i = 0$, and let step length $\eta_j = \Delta$;
if $\zeta^j(\eta_\infty) > 0$ **then**
  $\eta_j = \eta_\infty$; ▷ safeguard
else
  while $i < I$ **do**
    Let $s^* \in \operatorname{argmax}_{s \in \operatorname{ext}(\mathsf{EPM}_g)} s(\tilde{x} + r_x^j \eta_j)$;
    Compute a subgradient $\beta = r_t^j - s^* r_x^j$;
    if $\zeta^j(\eta_j) = 0$ **then**
      break;
    else if $\beta < 0$ **then**
      $\eta_j = \eta_j - \frac{\zeta^j(\eta_j)}{\beta}$; ▷ Newton step
    else
      $\eta_j = 2\eta_j$; ▷ bisection step
    $i = i + 1$;

$\eta'_j \geq 0$ such that, for all $\eta_j \in [0, \eta'_j)$, the algorithmic gradient $\nabla\zeta^j(\eta_j) > 0$; for all $\eta_j \in [\eta'_j, +\infty]$ (called the Newton step region), the algorithmic gradient $\nabla\zeta^j(\eta_j) \leq 0$.

After a finite number of bisection steps (at most $\lceil \log(\eta'_j/\Delta) \rceil$), the algorithm enters the Newton step region $[\eta'_j, +\infty]$, where the algorithmic gradient is always negative. Then, we prove that the algorithmic gradient $\nabla\zeta^j(\eta_j)$ at step i is different from that at step $i-1$, and the algorithm stays in the Newton step region. Since ζ^j is piece-wise linear (the number of its distinct algorithmic gradients is finite), the algorithm must terminate in a finite number of steps.

If at step $i-1$, $\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) = 0$, then the algorithm terminates at this step and finds the zero point. If at step $i-1$, $\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) < 0$, then we prove that $\nabla\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) \neq \nabla\zeta^j(\eta_j)$ and $\nabla\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) \leq 0$.

First, assume, to aim at a contradiction, that $\nabla\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) = \nabla\zeta^j(\eta_j)$. Knowing that the algorithmic gradient is monotone-decreasing, the piece-wise linearity of ζ^j implies that this algorithmic gradient is constant in the range $[\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}, \eta_j]$. It follows that for all $\delta \in [0, \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}]$, $\zeta^j(\eta_j - \delta) = \zeta^j(\eta_j) - \delta\nabla\zeta^j(\eta_j)$. Hence, $\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) = 0$, which leads to a contradiction.

Second, we show that $\nabla\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) \leq 0$. When $\frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)} \leq 0$, by the mononcity of $\nabla\zeta^j$, $\nabla\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) \leq \nabla\zeta^j(\eta_j) < 0$. When $\frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)} > 0$, as by assumption that $\nabla\zeta^j(\eta_j) < 0$, $\zeta^j(\eta_j)$ must be negative. Then, by the concavity of ζ^j, $\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) \leq \zeta^j(\eta_j) - \nabla\zeta^j(\eta_j)\frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)} = 0$. This implies that $\nabla\zeta^j(\eta_j - \frac{\zeta^j(\eta_j)}{\nabla\zeta^j(\eta_j)}) \leq 0$. □

From Prop. 6.5, the hybrid discrete Newton algorithm first executes bisection steps with increasing η_j and $\zeta^j(\eta_j)$. Then it enters into the Newton step region. After a single Newton step, $\zeta^j(\eta_j)$ becomes negative, and then monotonically increases to zero in a finite number of steps.

The discrete Newton algorithm in [157] is applied to the line search problem for submodular polyhedra, which are polars of extended polymatroids. In that context, it runs in a strongly

polynomial time. In our case, $\mathcal{C}$ contains the extended polymatroid, but it is unbounded in general. The corresponding line search problem may have no solutions, if the ray r^j is contained in the recession cone of $\mathcal{C}$. Therefore, Algorithm 6.1 needs a safeguard step, where we evaluate ζ^j at a user-defined infinity. One may also prove that Algorithm 6.1 runs in a strongly polynomial time, but a careful analysis for the unbounded case is needed.

6.4 Computational results

In this section, we conduct computational experiments to test the proposed cuts. The source code, data, and detailed results can be found in our online repository: github.com/lidingxu/Subcut.

Setup and performance metrics. The experiments are conducted on a server with Intel Xeon W-2245 CPU @ 3.90GHz and 126GB main memory. We use SCIP 8.0 [61] as a MINLP framework to solve the natural formulations of test problems. SCIP is equipped with `CPLEX` 22.1 as an LP solver, and `IPOPT` 3.14 as an NLP solver.

By Thm. 4.17, the simple lifted split $H_j := \{x \in \mathbb{R}^n : 0 \leq x_j \leq 1\} \times \mathbb{R}$ is a maximal $\mathrm{hypo}_{\mathcal{B}}(f)$-free set, where the splitting variable x_j is chosen as the most fractional entry of the relaxation solution. We have three settings of cut separation routines (cut separators). The *submodular cut* (resp. the *split cut*) setting adds intersection cuts derived from EE_f (resp. H_j), and the *default* setting does not add any intersection cuts. Our separators adhere to unified parameter settings that aim to maximize the likelihood of SCIP invoking our separators. These parameters for the cut separators in SCIP are detailed in [2]. Notably, during our experiments, we observed that the cut separators are predominantly influenced by the following parameters:

- SEPA_PRIORITY: the priority of the intersection cut separator. We set it to 100000 (the separators are called in a predefined order, which is given by the priorities of the separators).
- SEPA_DELAY: the default for whether the separation method should be delayed, if other separators found cuts. We set it to TRUE, i.e., delayed. (If the separator's separation method is marked to be delayed, it is only executed after no other separator found a cut during the price-and-cut loop).
- SEPA_MINVIOL: the minimal violation a cut must fulfill such that the cut can be added. We set it to 10^{-4}.
- SEPA_NCUTSLIMITROOT: the limit for the number of cuts generated at the root node. We set it to -1, meaning that the separation is unlimited.

Most cut separators in SCIP have priorities lower than 15, leading us to assign the highest priority to our cut separators. Consequently, SCIP calls our cut separators before the others during the optimization process.

The proposed cuts in this chapter are represented by the expression $\alpha x + \mu t \leq \beta$. When constructing a cut of this form to separate a point $(\tilde{x}, \tilde{t})$, it is considered numerically *ill-conditioned*, if the condition number $\max(\alpha, \mu)/\min(\alpha, \mu)$ becomes too large.

The objective of the proposed cuts is to approximate the constraint $f(x) \geq \ell t$, where f represents either a submodular function or an SS function. During our analysis, we observed that the magnitude of $f(x)$ can be significantly larger than 1. For submodular cuts, this

leads to a numerically ill-conditioned cut, where the magnitude of μ is much smaller than the magnitudes of the entries in α.

A similar issue arises in the numerical optimization of finite sums of nonlinear functions, such as problems of the form $\min g(x) := \min \sum_{j\in[k]} g_j(x)$. To enhance numerical stability during optimization, it is more favorable to optimize the average $g(x)/k$ rather than $g(x)$ itself. Therefore, we adopt a similar pre-processing step to scale our test problems.

Specifically, we scale the constraint $f(x) \geq \ell t$ into $f(x)/\chi \geq \ell t$, where χ represents a positive scaling factor. The purpose of this step is to ensure that the magnitude of μ becomes similar to that of α and β. The factor χ is selected as follows:

- For MAX CUT problems, χ is the number of edges of the graph.
- For PSEUDO BOOLEAN MAXIMIZATION problems, χ is the number of degree-4 monomials in the polynomial.
- For D-OPTIMAL DESIGN problems, χ is 1.

SCIP has internal routines of higher authority than any individual cut separator. These routines can control whether to invoke a cut separator and whether to apply the cuts found by the separator. Interfaces of these routines are not exposed publicly, but SCIP allows us to affect these routines through the parameters of cut separators. Therefore, we conduct the above three settings respectively in two distinct configurations: the *standalone* and the *embedded* configurations.

In the *standalone* configuration, we aim at measuring the performance of our cuts in a "clean" environment without interacting with other cuts, so we deactivate all of SCIP's internal cut separators. In the *embedded* configuration, we aim at measuring the performance of our cuts in a "real" environment. According to Example 6.10 of [100], our split cuts correspond to Gomory mixed integer cuts. To ensure a fair comparison, we require an equal level of implementation for intersection cuts, including the data structure and parameter settings. Hence, we replace SCIP's implementation of Gomory mixed integer cuts with our own implementation, thereby disabling SCIP's internal Gomory mixed integer cut separators in the embedded configuration.

We focus on the root node performance and measure the *closed root gap*. Let d_1 be the value of the first LP relaxation (without cuts added), let d_2 be the dual bound after all the cuts are added, and let p be a reference primal bound. The closed root gap $(d_2 - d_1)/(p - d_1)$ is the closed gap improvement of d_2 with respect to d_1. We also record the number of added cuts, the relative improvement to the default setting, and the total running time. For each configuration and setting, we compute these statistics' shifted geometric means (SGMs) with a shift of 1 over our test sets.

For each of the following experiments, we present and analyze computational results in the form of tables and scatter plots. The tables contain SGMs of the statistics, including the closed root gap (abbreviated as "closed"), the total running time (abbreviated as "time"), and the number of applied cuts (abbreviated as "cuts"). Moreover, the "relative" column displays the relative value of the closed root gap of one configuration that our cuts are enabled to that of the default configuration. Thus, the "relative improvement" due to our cuts is defined as the "relative" minus one. The scatter plots compare the closed root gap of each instance between two different settings. Furthermore, each scatter plot indicates the number of instances where one setting outperforms the other, referred to as "win" instances.

Experiment 1: MAX CUT. Consider an undirected graph $G = (V, E, w)$, where V is the set of nodes, E is the set of edges, and w is a weight function over E. For a subset S of V, its associated cut capacity is the sum of the weights of edges with one end node in S and the other end node in $V \setminus S$. The MAX CUT problem aims at finding a subset $S \subseteq V$ with maximum cut capacity. Let $V = [n]$, and we use a binary variable vector $x \in \mathcal{B}$ indicating whether vertices belong to S. The problem can be formulated as the following quadratic unconstrained binary optimization (QUBO) problem:

$$\max_{x \in \mathcal{B}} \sum_{\{i,j\} \in E} w_{ij}((1 - x_i)x_j + x_i(1 - x_j)).$$

When w is nonnegative, the cut capacity function (the objective function) is submodular.

The Biq Mac library [305] offers a collection of MAX CUT and QUBO instances of medium size. Our benchmark consists of two sub-benchmarks with 30 "g05" and respectively 30 "pw" MAX CUT instances with nonnegative weights from the library. These instances are generated randomly by Giovanni Rinaldi's `rudy` code [259, 263]. For each dimension $n = 60, 80, 100$, the "g05" sub-benchmark consists of 10 unweighted graphs with edge probability 0.5. For each graph density in $\{0.1, 0.5, 0.9\}$, the "pw" sub-benchmark consists of 10 graphs with integer edge weights chosen from $[0, 10]$.

The reference primal bounds are also from the Biq Mac library. We encode the hypograph reformulation (4.6) of the QUBO. SCIP will automatically reformulate the problem into a MILP via the reformulation-linearization technique (RLT) [6]. This MILP formulation is a special case of the extended formulation (6.8) of a degree-2 BMP with $m = 0$.

For the standalone configuration, the relative improvement of submodular cuts is 342% compared to 178% of split cuts. In the standalone configuration, we can compare the "clean" strengths of intersection cuts derived from different Boolean-hypograph-free sets. As observed from the scatter plots in Fig. 6.2, the submodular cut setting outperforms the split cut setting in 42 instances under the standalone configuration. Although split cuts are derived from maximal Boolean-hypograph-free sets and submodular cuts are derived from non-maximal ones, the clean performance of split cuts is worse. Regarding the embedded configuration, the relative improvement of submodular cuts is 85%, compared to 58% of split cuts. The scatter plot shows that the submodular cut setting surpasses the split cut setting in 34 instances under this configuration.

We observe that fewer split cuts are generated than submodular cuts. This means that the efficiency of some split cuts does not satisfy SCIP's internal criteria, so SCIP abandons more split cuts than submodular cuts. As two types of cuts are derived using the same principle but from different Boolean-hypograph-free sets, the distances between the relaxation points to the boundary of Boolean-hypograph-free sets determine the cut efficiency. This observation suggests that relaxation points are further from the boundary of the extended envelope epigraph than from the splits. The separation time of split cuts is shorter than that of submodular cuts, particularly for the "pw" instances with a high graph density (0.9). This is because separating submodular cuts requires solving nonlinear equations that involve sorting and computing graph cuts, while the split cuts can be computed in a closed form.

Experiment 2: PSEUDO BOOLEAN MAXIMIZATION. As mentioned, PSEUDO BOOLEAN MAXIMIZATION is a MUBO problem, a generalization of QUBO. We can use techniques from Sect. 4.1.3 to generate intersection cuts.

Configuration	Default		Submodular cut				Split cut			
	closed	time	closed	relative	time	cuts	closed	relative	time	cuts
standalone	0.026	4.33	0.111	4.418	22.9	215.48	0.075	2.78	7.04	75.04
embedded	0.097	4.77	0.161	1.852	68.19	162.5	0.139	1.575	9.4	67.6

Table 6.1 Summary of MAX CUT results

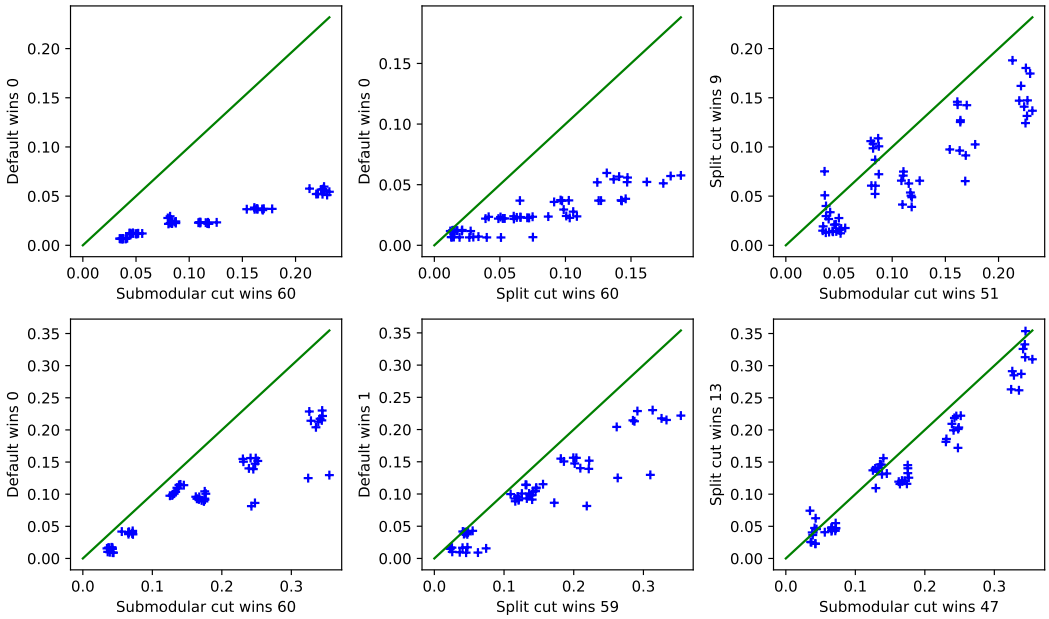


Figure 6.1 Scatter plots of MAX CUT results in standalone (top) and embedded (bottom) configurations

POLIP [245] is a library of polynomially constrained mixed-integer programming instances. All MUBO instances in POLIP with degree higher than 2 are 41 "autocorr_bern" instances, which are also included in MINLPLib [75, 300]. These instances arise from short ranged non-disordered lattice spin model (the Bernasconi model) [206] in theoretical physics. The problem is to determine a ground state in the Bernasconi model minimizing a degree-four energy polynomial: $\frac{n}{n-r+1}\sum_{i=0}^{n-r}\frac{1}{r(r-1)}\sum_{d=1}^{r-1}(\sum_{j=i}^{i+r-1-d} z_j z_{j+d})^2$, where $z \in \{-1,1\}^n$. The number n of variables in these instances is chosen from 20 to 60, and the interaction range r is chosen from 3 to 6. The problem is reformulated into a degree-4 BMP with $m = 0$ in MINLPLib through the transformation $z_j = 2x_j - 1$. SCIP constructs the extended formulation (6.8). We use the best-known primal bound from MINLPLib as the reference primal bound.

In Table 6.2, we report the computational results. For the standalone (resp. the embedded) configuration, the relative improvement of submodular cuts is 504% compared to 117% of split cuts. As indicated by the scatter plots in Fig. 6.2, the submodular cut setting outperforms the split cut setting in 29 instances under the standalone configuration. Regarding the embedded configuration, the relative improvement of submodular cuts is 98%, compared to 49% of split cuts. As indicated by the scatter plots, the submodular cut setting wins in 31 more instances than the spit cut setting under this configuration.

In both configurations, the submodular cuts are better than the split cuts in terms of the closed root gap. Moreover, under the embedded configuration, the difference in the relative

improvements between submodular cuts and split cuts is 48%. This is larger than 28% of MAX CUT benchmark under the same configuration. This divergence between degree-2 and degree-4 MUBO suggests that the submodular cuts are suitable for high-order Boolean multilinear constraints.

We recall that to solve the nonlinear equations, the hybrid discrete Newton algorithm needs oracle access to the value of the Boolean multilinear function. For some instances, a Boolean multilinear function may consist of thousands of multilinear terms. After a code timing analysis, we find that the separation of submodular cuts spends the most time computing the function value. Therefore, this is the main time performance bottleneck, which needs to be optimized in the future. In accordance with MAX CUT results, non-maximal $\mathcal{S}$-free sets may yield stronger cuts. Thus, the geometrical relation between the $\mathcal{S}$-free sets and the optimal tableau cone matters.

In Sect. 6.4.1, we conduct a branch-and-bound test.

Configuration	Default		Submodular cut				Split cut			
	closed	time	closed	relative	time	cuts	closed	relative	time	cuts
standalone	0.008	8.46	0.053	6.039	28.03	68.83	0.032	2.170	10.56	20.48
embedded	0.051	13.60	0.079	1.979	46.20	28.20	0.067	1.491	20.43	9.43

Table 6.2 Summary of PSEUDO BOOLEAN MAXIMIZATION results

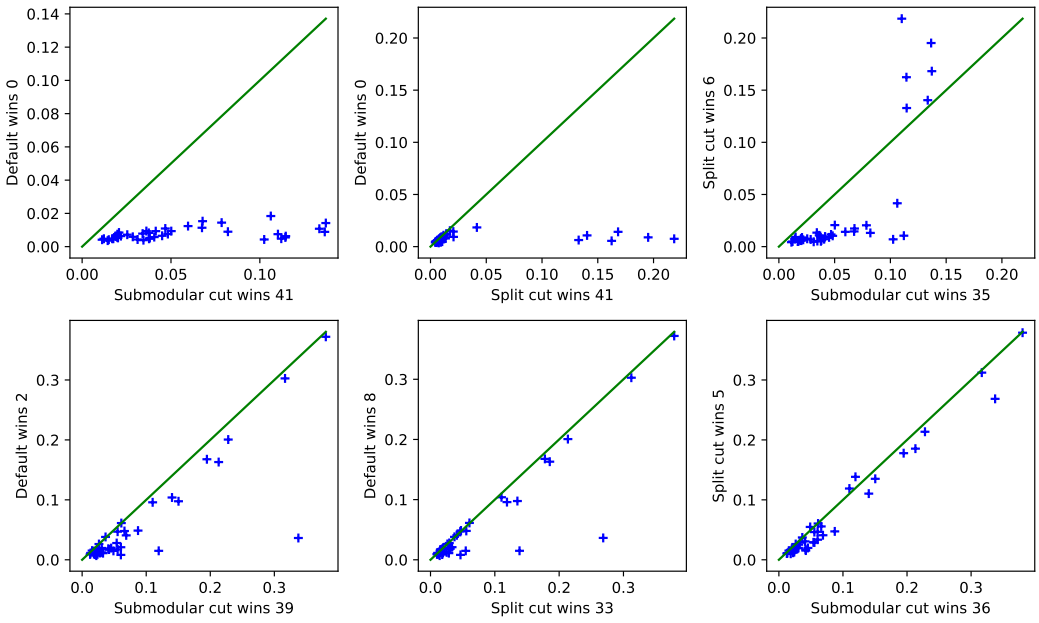


Figure 6.2 Scatter plots of PSEUDO BOOLEAN MAXIMIZATION results in standalone (top) and embedded (bottom) configurations

Experiment 3: BAYESIAN D-OPTIMAL DESIGN. As mentioned before, the BAYESIAN D-OPTIMAL DESIGN problem has a submodular maximization form (6.7). In particular, we can encode it as an extended formulation (6.8) in SCIP. SCIP generates gradient cuts for this convex MINLP. Therefore, we can obtain LP relaxations and simplicial conic relaxations.

Our benchmark consists of two sub-benchmarks. Recall that the binary vector x selects a subset of $\{M_j M_j^\top \in \mathbb{S}^m\}_{j\in[n]}$, and the information matrix $M(x)$ in (6.7) is the sum of matrices

in the selected subset. Thus, the problem data of (6.7) are variable dimension n, matrix size m, cardinality number k, and matrices $M_j M_j{}^\top$. We next outline the procedure for generating them.

The first sub-benchmark consists of 15 block design instances. We follow the method in [267] to generate these instances. Let $x := (x_{1,2}, x_{1,3}, \ldots, x_{1,m+1}, \ldots, x_{m,m+1}) \in \{0,1\}^n$, where $n = \binom{m+1}{2}$. Let $H(x)$ be an undirected graph with $m+1$ vertices. If $x_{i,i'} = 1$, there is an edge between vertices i, i'; otherwise, no edge connects them. We have $M(x) = PL(x)P^\top$, where $L(x) := \sum_{i,i'} x_{i,i'}(\mathbf{1}_i - \mathbf{1}_{i'})(\mathbf{1}_i - \mathbf{1}_{i'})^\top \in \mathbb{S}^m \subseteq \mathbb{R}^{m\times m}$ is the Laplacian of $H(x)$, and $P \in \mathbb{R}^{m\times(m+1)}$ is the matrix that transforms an $(m+1)$-dimensional vector v to the vector obtained by keeping the first m entries of v. In other words, $M(x)$ is the submatrix of the Laplacian of $H(x)$ obtained by removing its last row and last column. Then an optimal solution to (6.6) corresponds to the graphs with n nodes and k edges that have a maximum number of spanning trees. Note that $M_j = P(\mathbf{1}_i - \mathbf{1}_{i'}) \in \mathbb{R}^{m\times 1}$ is a single-column matrix, which is degenerated to a m-dimensional vector. We generate a block design instance for each combination of $m \in \{10, 11, 12\}, n = \binom{m+1}{2}, k \in \{m, m+1, m+2, m+3, m+4\}$. This results in a total of 15 combinations.

The second sub-benchmark consists of 30 random Gaussian instances. We generate a Guassian instance for each combination of

$$(n, m) \in \{(50, 20), (50, 30), (60, 24), (60, 36), (70, 28), (70, 42)\}$$

and

$$k \in \{m, m+1, m+2, m+3, m+4\}.$$

This results in a total of 30 combinations. We still let each M_j be a single-column matrix (i.e., a vector), and its entries are drawn from a Gaussian distribution with zero mean and a variance of $1/\sqrt{n}$.

We set the regularization constant ϵ to 10^{-6}. SCIP can find primal feasible solutions at the root node using its internal heuristics. We select the best primal bound given by these solutions among all settings as the reference primal bound. Since SCIP's internal gradient cuts are important for linearizing convex nonlinear constraints, we keep the gradient cuts but disable all integer-oriented cuts (GMI cuts and mixed-integer rounding cuts etc.) in the standalone configuration.

In Table 6.3, we report the computational results. We divide the results of block design and Gaussian random instances, since the density of matrices are different. Looking at the default setting in different benchmarks, there is no difference between the standalone and embedded configurations in terms of the closed root gap. This means that integer-oriented cuts do not improve the root node LP relaxations. We see the same problem for intersection cuts, which do not close the root gap but increase the computing time. In particular, the number of separated cuts is around one. Thereby, many intersection cuts are too weak to add to the cut pool.

We recall that intersection cuts and many integer-oriented cuts are LP-based cuts, i.e., derived from an LP relaxation of the extended formulation (6.8). Therefore, their strengths depend on the LP relaxation. Based on the types of MINLPs, there are two basic ways to construct initial LP relaxations. For nonconvex MINLPs, one way usually uses the factorable programming and term-wise envelopes [223]. Notable examples are Boolean multilinear constraints and continuous quadratic constraints [233]. The McCormick envelopes or Boolean

linearization techniques are used to construct their LP relaxations, which have a finite number of constraints.

For convex MINLPs, the other way linearizes nonlinear constraints, and the number of constraints in the LP relaxation can grow to infinity. This is because a convex nonlinear constraint is equivalent to an infinite number of linear constraints. Given that SCIP may incorporate numerous gradient cuts to approximate the convex MINLP (6.8), we can better understand its behavior in a simplified scenario. Consider a smooth convex body approximated by a polyhedral outer approximation, where each vertex and its associated faces define one or several simplicial cones, representing optimal tableau cones. As the polyhedron closely approximates the convex body, the vertex comes closer to the border manifold of the convex body, and the simplicial cones approach the tangent space of the manifold at that vertex. Consequently, the cones become very flat, and in the most extreme case, they turns into a hyperplane defining the tangent space. When a hyperplane intersects an $\mathcal{S}$-free set, this results in the hyperplane itself. Therefore, it is highly likely that our separators will generate weak intersection cuts. In summary, the weakness of intersection cuts is due to the flatness of the optimal tableau cone.

Benchmark	Configuration	Default		Submodular cut				Split cut			
		closed	time	closed	relative	time	cuts	closed	relative	time	cuts
Block design	standalone	0.59	20.46	0.59	1.0	18.71	1.84	0.59	1.0	11.62	1.77
	embedded	0.59	21.44	0.59	1.0	19.0	1.84	0.59	1.0	12.41	1.77
Gaussian	standalone	0.83	213.13	0.83	1.0	415.07	1.45	0.83	1.0	214.17	1.45
	embedded	0.83	214.77	0.83	1.0	426.33	1.45	0.83	1.0	214.14	1.45
All	standalone	0.75	98.47	0.75	1.0	149.54	1.57	0.75	1.0	82.6	1.55
	embedded	0.75	100.47	0.75	1.0	153.01	1.57	0.75	1.0	84.31	1.55

Table 6.3 Summary of BAYESIAN D-OPTIMAL DESIGN results

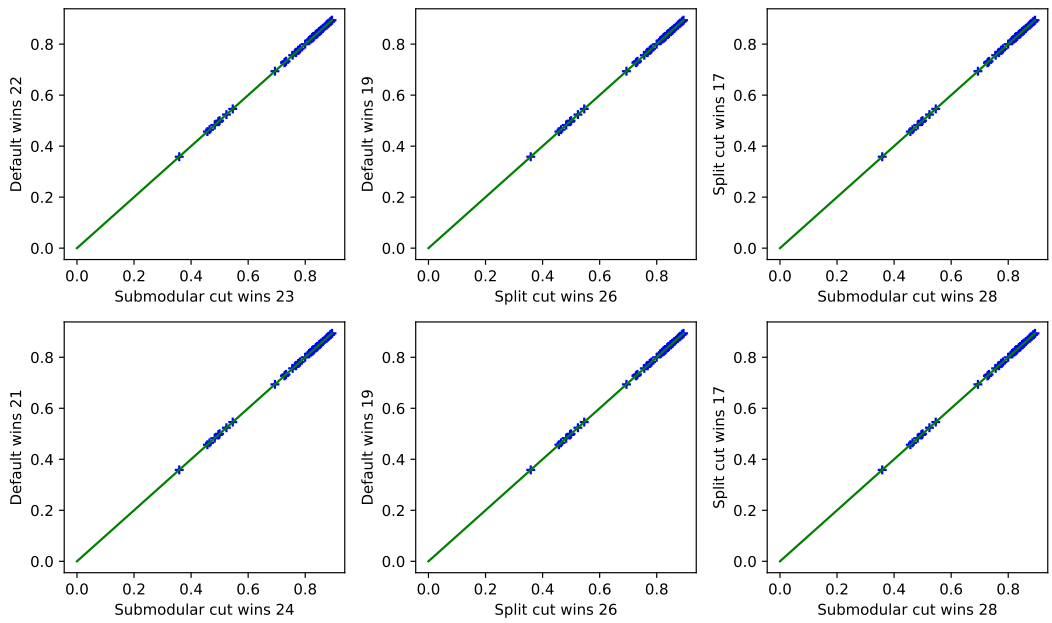


Figure 6.3 Scatter plots of BAYESIAN D-OPTIMAL DESIGN results in standalone (top) and embedded (bottom) configurations

6.4.1 Supplementary branch-and-bound computational results

We also have an additional branch-and-bound test for MAX CUT problem instances in Sect. 6.4. This test was designed to assess the performance and properties of our cuts in a "production-level" environment, which presents more complex challenges compared to the root node experiment. As such, the parameter settings and analysis in this test are more intricate and require a detailed explanation, which we provide below.

We conducted our tests under the embedded configuration, where the branching rule, node selection rule, and primal heuristics adhere to SCIP's defaults. We made adjustments to some parameters specifically to control the behavior of our cut separators in the branch-and-bound algorithm. These parameters are as follows:

- SEPA_FREQ: the default frequency for separating cuts. We set it to 0, meaning that our cut separators are called at the root node.
- SEPA_NCUTSLIMITROOT: the limit for the number of cuts generated at the root node. We set it to 60.
- SEPA_MAXBOUNDDIST: the default maximal relative distance from the current node's dual bound to primal bound compared to the best node's dual bound for applying separation. We set it to 1, meaning that separation is applied at all search nodes.

Due to the substantial number of parameter combinations, tuning the parameters for the branch-and-bound test is more challenging compared to the root node experiment. For instance, SCIP's internal Gomory mixed-integer cut separator [1, 108] is limited to applying at most 30 cuts at the root node, while SCIP's quadratic intersection cut separator [3, 89] employs at most 20 cuts at the root node and 2 cuts at each non-root node.

In a preliminary branch-and-bound test, we find that even the default setting can solve the "pw" instances of density 0.1 within 100 seconds, while all settings run 3600 seconds on the other instances. To have an unbiased result, we remove "pw" instances of density 0.1 and create a sub-benchmark called MAX CUT-sub.

For the following branch-and-bound test, we measure the closed duality gap (abbreviated as gap), the relative improvement of the closed duality gap to the default setting, the number of search nodes (abbreviated as nodes), and the number of applied cuts.

Benchmark	Default		Submodular cut				Split cut			
	gap	nodes	gap	relative	nodes	cuts	gap	relative	nodes	cuts
MAX CUT-sub	0.605	231372	0.596	0.981	220176	59.87	0.618	1.026	207078	42.2

Table 6.4 Summary of MAX CUT-sub results in the embedded branch-and-bound test

Our observations indicate that the submodular cut setting performs slightly worse than the default setting, while the split cut setting performs marginally better than the default setting. As shown in Fig. 6.4, the difference in closed duality gaps between the submodular/split cut and default settings is no more than 2%. This shows that the optimization landscape of MAX CUT problems is very complicated. As for our parameter settings, intersection cuts cannot have a significant impact on the branch-and-bound algorithm.

In contrast to the results in the root node experiment, we find that the split cut setting outperforms the submodular cut setting in the branch-and-bound test. Detailed cut information obtained during debugging reveals that the condition number of submodular cuts can be

thousands of times larger than that of split cuts, and the submodular cuts can be denser as well. Consequently, these numerical properties make the submodular cuts less stable and efficient compared to the split cuts.

When considering the approximation of the Boolean-hypograph $\mathrm{hypo}_{\mathcal{B}}(g)$, where g represents any function over $\mathcal{B}$, we can deduce from Thm. 4.17 that the splits define a class of maximal Boolean-hypograph-free sets. Although the split cuts are independent of the values of g, we can use the split cuts to approximate the Boolean-hypograph of g. While one can find other Boolean-hypograph-free sets based on the values of g, the resulting cuts will likely exhibit the same numerical properties as our submodular cuts.

As a result, future research should consider this finding when exploring intersection cuts. However, it is also worthwhile to investigate the performance of submodular cuts in other problems and algorithms, such as PSEUDO BOOLEAN MAXIMIZATION problems and the diving heuristic.

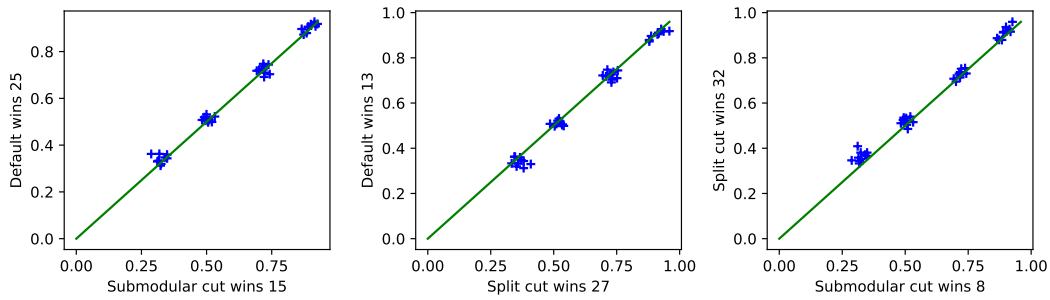


Figure 6.4 Scatter of MAX CUT-sub results in the embedded branch-and-bound test

6.5 Conclusion

We construct Boolean-hypograph-free sets for submodular functions. Our construction relies on a new continuous extension of submodular functions. We characterize maximal Boolean-hypograph-free sets, and generalize our results to sets involving submodular-supermodular functions. These yield intersection cuts for Boolean multilinear constraints. We exploit the submodular structure in an extended formulation of the D-OPTIMAL DESIGN problem. We propose a hybrid discrete Newton algorithm that can compute intersection cuts efficiently and exactly. The computational results show that intersection cuts derived from the submodularity are better than those derived from split cuts for MAX CUT and PSEUDO BOOLEAN MAXIMIZATION problems in the root-node experiments. For convex MINLPs, our computational results on the BAYESIAN D-OPTIMAL DESIGN problem suggest that simplicial conic relaxations given by gradient cuts can be flat, which makes intersection cuts weak.

Chapter 7

Branch-and-price for submodular bin packing

7.1 Introduction

Bin packing (BP) is an important combinatorial optimization problem with applications in various fields, including call centers, healthcare, container shipping, and cloud computing. These applications are typically modeled as BP problems that aim to pack unsplittable items into a minimum number of bins, with a capacity constraint on each bin. Formally, a BP problem can be written as the following Binary Linear Programming (BIP) problem:

$$\min \quad \sum_{j\in\mathcal{M}} y_j, \tag{7.1a}$$

$$s.t. \quad \sum_{i\in\mathcal{N}} \mu_i v_{ij} \le c y_j, \qquad \forall j \in \mathcal{M}, \tag{7.1b}$$

$$\sum_{j\in\mathcal{M}} v_{ij} = 1, \qquad \forall i \in \mathcal{N}, \tag{7.1c}$$

$$v_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{N}, j \in \mathcal{M}, \tag{7.1d}$$

$$y_j \in \{0,1\}, \qquad \forall j \in \mathcal{M}, \tag{7.1e}$$

where $\mathcal{M} := \{1, \cdots, m\}$ is the index set of potential bins (m is the number of potential bins), $\mathcal{N} := \{1, \cdots, n\}$ is the index set of items (n is the number of items), c is the capacity which is the same for every bin, and μ_i is the size of item i. Variable y_j decides whether bin j is used, and variable v_{ij} indicates whether item i is allocated to bin j. Capacity constraints (7.1b) stipulate that the capacities of bins are not exceeded, and set partition constraints (7.1c) require that each item is exactly allocated to one bin.

In many practical applications of BP, nominal item sizes μ are not revealed before the allocation decision is made, so uncertainty arises. Probabilistic modeling of capacity constraints (7.1b) allows item sizes μ to be random parameters, and thus the uncertainty is taken as a probability distribution on μ. We consider two commonly used probabilistic BP models. The first probabilistic model is the BP with chance constraints (BPCC) [280]. By assuming item sizes μ following a given (multivariate) probability distribution, BPCC requires that each capacity constraint in (7.1b) should be respected with a probability at least α, written as the

following chance constraints [85]:

$$\mathbb{P}(\sum_{i\in\mathcal{N}} \mu_i v_{ij} \leq c y_j) \geq \alpha, \qquad \forall j \in \mathcal{M}. \tag{7.2}$$

The second probabilistic model is the distributionally robust BP (DRBP) [319, 97]. It models the worst case of chance constraints [153]. More specifically, given a family $\mathscr{D}$ of probability distributions of μ, DRBP requires that each chance constraint in (7.2) should be respected for any probability distribution within $\mathscr{D}$. Thereby, capacity constraints of DRBP can be formulated as the following distributionally robust constraints

$$\inf_{\mu\sim\mathscr{D}} \mathbb{P}(\sum_{i\in\mathcal{N}} \mu_i v_{ij} \leq c y_i) \geq \alpha, \qquad \forall j \in \mathcal{M}. \tag{7.3}$$

Computational optimization of BPCC and DRBP models is challenging due to probabilistic constraints. Stochastic optimization methods can tackle mathematical optimization problems with probabilistic constraints. The sample average approximation (SAA) is a common stochastic optimization method for chance-constrained and distributionally robust optimization problems [216, 60]. It approximates these problems as two/multi-stage MILP problems and computes approximate solutions that converge to an optimal solution in a probabilistic sense. Previous works, such as [319, 123, 41], apply tailored SAA methods to solve BPCC and DRBP.

Several recent works show that, under various assumptions on probabilistic distributions, BPCC and DRBP are equivalent to or well-approximated by a deterministic optimization problem, namely, submodular BP (SMBP). It is shown in [97] that, BPCC has an SMBP formulation, if item sizes μ follow independent Gaussian distributions; SMBP also provides an upper bound for BPCC with item sizes μ under general independent distributions over bounded intervals (we note that then SMBP becomes a restriction of BPCC, and thus its solution is always feasible to BPCC.). It is shown in [318] that DRBP has an SMBP formulation, if distributions in $\mathscr{D}$ have the same mean values and the same diagonal covariance matrix.

Given the applicability of the previous assumptions, SMBP is an appealing alternative formulation to BPCC and DRBP, as it can be solved optimally in a finite time, while the convergence rate of SAA methods for BPCC and DRBP depends on the number of samples. SMBP already finds its applications in cloud computing [97], surgery planning [121], and operating room planning [301]. The environment is highly dynamic for these applications, and uncertainty plays a significant role in practical models. These applications give rise to a need for efficient algorithms to solve SMBP.

In this chapter, we study the exact algorithms for solving SMBP. SMBP has the following Binary Nonlinear Programming formulation:

$$\min \quad \sum_{j \in \mathcal{M}} y_j, \tag{7.4a}$$

$$s.t. \quad \sum_{i \in \mathcal{N}} a_i v_{ij} + \sigma \sqrt{\sum_{i \in \mathcal{N}} b_i v_{ij}} \leq c y_j, \qquad \forall j \in \mathcal{M}, \tag{7.4b}$$

$$\sum_{j \in \mathcal{M}} v_{ij} = 1, \qquad \forall i \in \mathcal{N}, \tag{7.4c}$$

$$v_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{N}, j \in \mathcal{M}, \tag{7.4d}$$

$$y_j \in \{0,1\}, \qquad \forall j \in \mathcal{M} \tag{7.4e}$$

where a_i, b_i are parameters inferred from the distribution of μ_i. This formulation is a compact nonlinear version of (7.1). We remark that the left-hand side of the constraint (7.4b) is a submodular function over x [21], so SMBP is named after this function. A constraint in the form of (7.4b) with y_j fixed to 1 is called a submodular knapsack constraint.

To solve SMBP, the state-of-art exact algorithm uses general-purpose integer-programming solvers to solve its Binary Second-Order Conic Programming (BSOCP) reformulation [318], which valid inequalities can further strengthen [21]. The experiment in [318] shows that small instances with item number n up to 40 and bin numbers m up to 10 can be solved to optimality by this exact algorithm.

The intuition underlying this chapter is that the decomposition is a promising approach to tackling large-scale classical BPs: a BP is reformulated into a set cover formulation based on enumerating all feasible packing patterns; then its continuous relaxation is solved using a column generation approach [154]. The branch-and-price algorithm integrates column generation with the branch-and-bound algorithm. It is the state-of-the-art exact algorithm for solving DW decomposition of classical BPs [304, 120].

We propose the first DW decomposition and set cover formulation for SMBP, and design a branch-and-price algorithm with tailored methods for solving pricing problems. After our DW decomposition of SMBP, the nonlinearity moves to the pricing submodular knapsack subproblem, which has a linear objective function and a submodular capacity constraint. One can avoid the growing number of nonlinear constraints (7.4b) in the compact formulation (7.4), when solving larger instances.

The DW decomposition provides a skeleton of our main algorithm. The techniques for solving general DW decomposition problems are vast, to name a few, we refer to [141] for stabilization techniques, [103] for lexicographic pricing, [303] for goal cuts and early termination, and [184] for non-robust cuts. In [250], a simple parameterization enables the use of several advanced techniques in the branch-cut-and-price VRPsolver [252]: automatic stabilization by smoothing [251], limited-memory rank-1 cuts [247], enumeration, hierarchical strong branching over accumulated resources [150], and limited discrepancy search diving heuristics. In this chapter, we focus on algorithmic innovation that exploits the specific nonlinear structure of pricing problems and new techniques to speed up the convergence of column generation.

As the study in [318] for the compact formulation, the nonlinearity is a crucial feature for model representability, so it is unavoidable and needs a special algorithmic treatment. In our case, pricing problems have submodular knapsack constraints involving nonlinear functions. We give two different views of nonlinearity. First, we can represent the submodular knapsack

constraint via second-order constraints, which many general solvers then accept. Alternatively, we propose a non-convex Mixed-Binary Quadratically Constrained Programming (MBQCP) formulation for the submodular knapsack. A PWL function is linear in each partition of its domain and can be modeled by a MILP formulation [299]. PWL functions have been used to approximate or relax non-convex MINL) problems [149]. Despite its non-convexity, the critical feature of the MBQCP formulation is that its only nonlinear function is a univariate quadratic function, which is easy to approximate using a PWL function. We construct the PWL relaxation for the submodular knapsack and combine it with cutting planes to form an exact PWL relaxation-based branch-and-cut (PWL-B&C) algorithm.

The submodular knapsack is essential as it models the chance-constrained knapsack problem [162]. We thus provide an approach for solving submodular knapsack problems different from the pure valid inequality approach in [21, 22].

We propose several strategies to accelerate the convergence of the branch-and-price algorithm, i.e., improve primal and dual bounds. The Farley bound [136, 294] is an early valid dual bound before the termination of the column generation procedure [304, 155]. The formula for the Farley bound imposes a condition on whether an exact pricing algorithm can improve the current dual bound. If the condition is not satisfied, the exact pricing algorithm is unnecessary, so we can use fast pricing heuristic. Our branch-and-price algorithms use a hybrid pricing strategy to speed up the column generation procedure. The hybrid pricing strategy is thus an intermediate between exact pricing and heuristic pricing strategies [67].

There are few publicly available instances of SMBP problems. In [97], there is a method to generate instances from BPCC and DRBP under various distributions. We generate instances of three different scales by this method and conduct computational experiments on them. We implement our branch-and-price algorithm for the set cover formulation of SMBP and find that it outperforms existing methods, which solve the compact BSOCP formulation [318]. Our core innovation, PWL-B&C pricing algorithm, and hybrid pricing strategy, significantly improve the branch-and-price algorithm.

7.1.1 Literature review

As mentioned, there are several steps of transformation from BP with uncertainty to SMBP. We review these transformations and algorithms for solving associated transformed models.

The surgery planning problem is a typical application of BP with uncertainty in healthcare, where the surgery duration (item size) is assumed to be stochastic. Some pioneering works [123, 41] allow violations to capacity constraints (the left-hand side of (7.1b) thus can be greater than the capacity) rather than consider chance constraints. To minimize these violations, they use a penalty approach by adding the expectation of the sum of these violations into the objective function. Therefore, the transformed BP model is a standard stochastic optimization problem called stochastic BP (SBP). SBP can be further modeled and solved as a stochastic two-stage mixed-integer programming problem: the first stage variables are the bin variables y, the second stage variables are the item variables v, and the expected violation is the second stage objective. In some works [79, 122], only the expected penalty is considered in the models.

Compared to SBP, BPCC, and DRBP can control the violation of each capacity constraint with a guaranteed probability bound, and thus they are more accurate models for BP with uncertainty. To solve BPCC and DRBP, there are approximation algorithms and exact algo-

rithms. (Sampling-based) approximation algorithms usually converge asymptotically to an optimal solution when the sampling number increases (as SAA methods), and exact algorithms usually converge in finite time. However, exact algorithms are mostly available for deterministic optimization problems.

In [279], a variant of BPCC in surgery planning is studied: items (surgeries) are allocated to a given set of bins (time blocks), the goal is to minimize the sum of expected capacity residuals (undertime), subject to chance constraints for overuse of bins' capacities (overtime). Assuming that the operation duration follows a multivariate normal distribution, the authors reformulate the problem as a deterministic optimization problem containing a convex objective and submodular capacity constraints. In [280], a special BPCC with the probability distribution over finite support is studied. It has a BIP formulation, which an exact algorithm can solve. As mentioned before, for various probabilistic distributions [97, 318], BPCC and DRBP admit a deterministic SMBP reformulation, which can be solved by exact algorithms.

Regarding solution algorithms, an SAA-based algorithm [319] can solve BPCC approximately, whose scenario subproblem is solved exactly by a DW decomposition approach. In addition, SMBP reformulation of DRBP [319] can be approximated by a MILP, which is solved by a DW decomposition approach. The only tailored exact algorithm for BPCC and DRBP [318] solves their compact SMBP reformulations. As for the deterministic variant of BPCC, the authors of [279] propose an exact outer approximation algorithm enhanced with PWL relaxation of submodular knapsack constraints, and the algorithm is a multi-search tree method, i.e., an underlying MILP solver will be called multiple times. In conclusion, no exact algorithm based on DW decomposition exists to solve SMBP. Meanwhile, DW decomposition is already used for various approximated problems.

This exception may be due to the lack of efficient exact algorithms to solve submodular knapsack problems. We note that in [319], DW decomposition is *de facto* applied to a MILP problem, and thus the pricing problems are also MILPs such as classical knapsack problems [76], which can be solved efficiently by general-purpose integer programming solvers. The efficiency of general solvers is mostly due to the lifted cover inequalities, which are strong valid inequalities for knapsack polytope and can be constructed via sequence-independent lifting [163]. However, for submodular knapsack, the computation of lifted cover inequalities is not tractable [22]. As we know from the literature, the tailored algorithm can be much better than general solvers for many variants of classical knapsack, because these algorithms can exploit more problem structures than general solvers. To name a few, we refer to the quadratic knapsack [78, 145], the multidimensional knapsack [256], and the quadratic multi-knapsack [53, 242].

Although general solvers are almost as complex as a black box for users, we can at least understand how they solve the submodular knapsack. The submodular knapsack can be reformulated as a BSOCP problem, which is an acceptable formulation to `CPLEX` [70] and `SCIP` [59]. These solvers implement LP outer approximation-based branch-and-cut (LP-B&C) algorithm [96] to solve the BSOCP or general Mixed-Integer Second-Order Conic Programming (MISOCP) problems. The LP outer approximation is sometimes called the polyhedral outer approximation (or polyhedral relaxation). In fact, any second-order conic program (SOCP) is polynomially reducible to a linear program [51]. As for a submodular knapsack constraint, general solvers will linearize it into an intersection of a set of classical knapsack constraints, thus, inefficiency arises if too many linearizations are applied.

On the other hand, there are alternative exact approaches to solve some classes of nonconvex MINLPs using PWL relaxations [113, 149]. For example, [113] obtains a convex MINLP relaxation for nonconvex MINLPs with separable nonconvex functions. The authors distinguish between convex and concave parts and then convexify the concave parts by PWL functions.

Regarding the submodular knapsack, its BSOCP (a convex MINLP) formulation has a nonlinear function over all the problem variables, which is difficult to approximate when the dimension is high. On the other hand, the nonconvex MBQCP formulation, where the only nonlinear function is a univariate quadratic function on a slack variable. The resulting PWL relaxation in the experiment is stronger than pure polyhedral relaxation. In our case, we will show that the quadratic function can be approximated in a “dimension-free" way, since the nonlinearity is concentrated on a single variable. [279] only uses PWL relaxations to approximate the submodular knapsack. It requires refining PWL relaxations to achieve convergence, so their multi-search tree algorithm needs to restart the MILP solver from scratch in each iteration. In contrast, we prefix PWL relaxations and use cutting planes to achieve convergence. Therefore, our single-search tree does not need to restart the MILP solver.

We look at the recent development of DW decomposition and branch-and-price algorithm for solving MINLPs (see [12]), such as recursive circle packing (RCP) problems [155], binary quadratic problems [81], and facility location with general nonlinear facility cost functions [241]. There may be several ways to divide a MINLP into master and subproblems, so a MINLP may admit different DW decompositions. In [81], the authors study the strengths of different DW decompositions for binary quadratic problems. In most cases, after applying the DW decomposition to the compact MINLP formulation, the master problem is a MILP, and the pricing problems are MINLPs. Since pricing problems are solved in thousands of iterations, [155] shows that any improvement in the pricing algorithm can speed up the convergence of column generation.

7.1.2 Contribution

In summary, our contribution in this chapter is threefold. As far as we know, the previous work applies DW decomposition for an approximated MILP for SMBP, and thus nonlinearity is not considered in the solving process. So, we are the first to apply DW decomposition for SMBP with the nonlinearity considered. Built on the basic DW decomposition and branch-and-price algorithm, we develop a new hybrid pricing strategy technique to speed up the column generation, which can avoid computationally expensive exact pricing while not worsening the dual bound. Second, for pricing submodular knapsack problems, we propose a new MBQCP formulation and its PWL relaxation, and design a new PWL-B&C algorithm as an alternative exact algorithm to the conventional LP-B&C algorithm, which is based on valid inequalities. Finally, we perform computational experiments on many instances to evaluate the proposed algorithms. The computational results show that our tailored branch-and-price algorithms for DW reformulation outperform the conventional branch-and-cut algorithm for BSOCP formulation implemented in a state-of-art commercial solver; and the PWL-B&C algorithm can be a standalone algorithm for submodular knapsack. The source code and benchmark are released on our project website https://github.com/lidingxu/cbp.

7.1.3 Outline of the chapter

This chapter is organized as follows. In Sect. 7.2, we describe the set cover formulation of SMBP. In Sect. 7.3, we introduce the critical components of our branch-and-price algorithm: the branching rule, column generation, dual bound computation, initial columns, and primal heuristics. In Sect. 7.4, focusing on solving the pricing problem, we present the pricing heuristic, reformulations of the pricing problem, PWL relaxation, the exact pricing algorithm, and the hybrid pricing strategy. In Sect. 7.5, we show the computational results of the proposed algorithms for instances generated from the literature and analyze their performance. In Sect. 7.6, we end this chapter with a conclusion and future research directions.

7.2 Set cover formulation

In this section, we propose a new set cover formulation for SMBP. The formulation is derived similarly to the DW decomposition of the classical linear BP [120]. This formulation can be solved efficiently by a branch-and-price algorithm.

A column p is defined by a binary vector as $(d_{1p}, d_{2p}, \ldots, d_{np})$, where $d_{ip} = 1$ if item i is contained in the column p. A column is called *feasible* if the combination of its items can fit into a bin, i.e., satisfies the submodular capacity constraint (7.4b). The set cover formulation is based on enumerating all feasible columns, the number of which can be exponential to the number of items.

Set notation:

- $\mathcal{P}$: the set of all feasible columns.

Decision variables:

- $\lambda_p = \begin{cases} 1, & \text{if column } p \text{ is used by the solution} \\ 0, & \text{otherwise} \end{cases}$ for $p \in \mathcal{P}$.

We obtain the following set cover formulation for SMBP:

$$\min \quad \sum_{p \in \mathcal{P}} \lambda_p, \tag{7.5a}$$

$$s.t. \quad \sum_{p \in \mathcal{P}} d_{ip} \lambda_p \geq 1, \qquad \forall i \in \mathcal{N}, \tag{7.5b}$$

$$\lambda_p \in \{0, 1\}, \qquad \forall p \in \mathcal{P}. \tag{7.5c}$$

The set cover constraint (7.5b) specifies that each item i ($i \in \mathcal{N}$) is contained in at least one bin. The set cover reformulation already finds applications in vehicle routing [247] and unsplittable multi-commodity flows [313].

The compact formulation (7.4) is a MINLP, but the set cover formulation (7.5) is a MILP. Moreover, the number of nonlinear constraints in the compact formulations equals the number of potential bins. The nonlinearity of the set cover formulation is *de facto* 'hidden' in the pricing subproblems, and each pricing subproblem has only one nonlinear constraint.

Remark 7.1. *(Modeling of BSOCP constraints) We give a way to obtain a BSOCP formulation of the constraint* $\sum_{i \in \mathcal{N}} a_i v_{ij} + \sqrt{\sum_{i \in \mathcal{N}} b_i v_{ij}} \leq d$*, where* $d = cy_i$ *or* $d = c$*. Since* $v_{ij} \in \{0, 1\}$*, the*

square root $\sqrt{\sum_{i\in\mathcal{N}} b_i v_{ij}}$ *equals* $\sqrt{\sum_{i\in\mathcal{N}} b_i v_{ij}^2}$*. Then the constraint is equivalent to a SOCP constraint* $\sqrt{\sum_{i\in\mathcal{N}} b_i v_{ij}^2} \le d'$*, where* $d' := d - \sum_{i\in\mathcal{N}} a_i v_{ij}$*. Then one can further reformulate the SOCP constraint into several 3d SOCP constraints* $x_1 \ge \sqrt{x_2^2 + x_3^2}$*, which is acceptable by* CPLEX *[71].*

We can obtain a BSOCP formulation of SMBP (7.4) through the above discussion. When comparing two formulations, a formulation is said to be "stronger", if it yields a better dual bound.

Proposition 7.2. *The linear relaxation of the set cover formulation* (7.5) *is stronger than the continuous SOCP relaxation of the BSOCP formulation of* (7.4)*.*

Proof. Let

$$F_j := \{(v_{1j}, \cdots, v_{nj}, y_j) \in \{0,1\}^{n+1} : \sum_{i\in\mathcal{N}} a_i v_{ij} + \sigma\sqrt{\sum_{i\in\mathcal{N}} b_i v_{ij}} \le c y_j\}$$

be the feasible set of the $j-$th constraint in the BSOCP formulation of (7.4). Therefore, the feasible set of the BSCOP formulation is $F = \prod_{j\in\mathcal{M}} F_j$.

Let $\overline{F}_j$ be the continuous relaxation of F_j, and

$$\overline{F}_j = \{(v_{1j}, \cdots, v_{nj}, y_j) \in [0,1]^{n+1} : \sum_{i\in\mathcal{N}} a_i v_{ij} + \sigma\sqrt{\sum_{i\in\mathcal{N}} b_i v_{ij}} \le c y_j\}.$$

Therefore, the feasible set of the continuous relaxation of the BSCOP formulation is $\overline{F} = \prod_{j\in\mathcal{M}} \overline{F}_j$.

On the other hand, the points of F_j are zero vectors and $(p, 1)$ ($p \in \mathcal{P}$). Therefore, its convex hull is

$$\mathrm{conv}(F_j) = \{(v_{1j}, \cdots, v_{nj}, y_j) \in [0,1]^{n+1} : \exists \lambda_p \in [0,1]^{\mathcal{P}} \wedge \sum_{p\in\mathcal{P}} \lambda_p = y_j \wedge v = \sum_{p\in\mathcal{P}} d_p \lambda_p\}.$$

We note that $\overline{F}_j$ is also a convex relaxation of F_j, hence $F_j \subset \mathrm{conv}(F_j) \subset \overline{F}_j$.

The optimum of the continuous relaxation of the BSOCP formulation is

$$\min_{(v,y)\in\overline{F}, v \text{ satisfies (7.4c)}} \sum_{j\in\mathcal{M}} y_j.$$

An optimal solution of the LP relaxation of the set cover formulation satisfies $\sum_{p\in\mathcal{P}} d_{ip}\lambda_p = 1$ ($i \in \mathcal{N}$), and the optimal value is exactly the same as $\min_{(v,y)\in\prod_{j\in\mathcal{M}}\mathrm{conv}(F_j), v \text{ satisfies (7.4c)}} \sum_{j\in\mathcal{M}} y_j$. Since $\prod_{j\in\mathcal{M}} \mathrm{conv}(F_j) \subset \overline{F}$, the result follows. □

7.3 Branch and price

Solving the set cover formulation with an exponential number of binary variables is challenging. In this section, we present an exact branch-and-price algorithm to solve the set cover

formulation of SMBP. The branch-and-price algorithm integrates column generation with the branch-and-bound algorithm to solve the LP relaxation efficiently. In the following subsections, we describe the important steps of our branch-and-price algorithm: the branching rule, column generation, primal heuristics, and dual bound computation.

7.3.1 Branching rule

Our branch-and-price algorithm uses the Ryan/Foster branching rule [264]. The branching rule selects a pair of items $i_1 \in \mathcal{N}$ and $i_2 \in \mathcal{N}$ that must either be packed together or not packed together. We denote by

- $\mathcal{S}$: the set of item pairs that are forced to be packed together such that, if a column p respects $\mathcal{S}$, then for $(i_1, i_2) \in \mathcal{S}$, $d_{i_1 p} = d_{i_2 p}$;
- $\mathcal{D}$: the set of item pairs that are not allowed to be packed together such that, if a column p respects $\mathcal{D}$, then for $(i_1, i_2) \in \mathcal{D}$, $d_{i_1 p} + d_{i_2 p} \leq 1$.

Indeed, $(\mathcal{S}, \mathcal{D})$ exactly describes the branching decisions made for each node of the search tree, whose nodes are constructed and selected by `SCIP`'s internal rules [5] in our implementation. We denote by

$$\mathcal{P}_{\mathcal{S},\mathcal{D}} := \{p \in \mathcal{P} \,|\, \forall (i_1, i_2) \in \mathcal{S}\ d_{i_1 p} = d_{i_2 p} \wedge \forall (i_1, i_2) \in \mathcal{D}\ d_{i_1 p} + d_{i_2 p} \leq 1\}$$

the set of feasible columns respecting branching constraints induced by $(\mathcal{S}, \mathcal{D})$. We refer to $\mathcal{P}_{\mathcal{S},\mathcal{D}}$ as the $(\mathcal{S}, \mathcal{D})$-feasible columns.

At each node of the search tree, the set cover problem (7.5) is restricted to the branching decision set $(\mathcal{S}, \mathcal{D})$, i.e., it follows as

$$\min \quad \sum_{p \in \mathcal{P}_{\mathcal{S},\mathcal{D}}} \lambda_p, \tag{7.6a}$$

$$s.t. \quad \sum_{p \in \mathcal{P}_{\mathcal{S},\mathcal{D}}} d_{ip} \lambda_p \geq 1, \qquad \forall i \in \mathcal{N}, \tag{7.6b}$$

$$\lambda_p \in \{0, 1\}, \qquad \forall p \in \mathcal{P}_{\mathcal{S},\mathcal{D}}. \tag{7.6c}$$

The above problem (7.6) is called the *master problem*, and its LP relaxation is called the *master LP problem*.

Given a solution λ of the LP relaxation, if λ is not integral, the branching rule chooses an item pair to branch. It first creates an n-by-n matrix, and computes its entries as $M_{i_1 i_2} = \sum_{p \in \mathcal{P}_{\mathcal{S},\mathcal{D}} : d_{i_1 p} = d_{i_2 p} = 1} \lambda_p$ for all $i_1, i_2 \in \mathcal{N}$. Since, for an integral solution, $M_{i_1 i_2}$ must be either 0 or 1, the branching rule chooses the most fractional entry (i'_1, i'_2) such that $i'_1, i'_2 = \operatorname{argmin}_{i'_1, i'_2 \in \mathcal{N}} |0.5 - M_{i'_1 i'_2}|$. Then, the rule adds (i'_1, i'_2) to $\mathcal{S}, \mathcal{D}$, respectively.

7.3.2 Column generation

We present a column generation method to solve the master LP problem.

The column generation procedure starts with a subset of $(\mathcal{S}, \mathcal{D})$-feasible columns of the master LP problem, adds columns, and solves the restricted LP iteratively. Given a subset

$\mathcal{P}'_{\mathcal{S},\mathcal{D}}$ of $\mathcal{P}_{\mathcal{S},\mathcal{D}}$, the corresponding restricted LP problem, namely the *Restricted Master LP* (RMLP) problem, is

$$\min \quad \sum_{p\in\mathcal{P}'_{\mathcal{S},\mathcal{D}}} \lambda_p, \tag{7.7a}$$

$$s.t. \quad \sum_{p\in\mathcal{P}'_{\mathcal{S},\mathcal{D}}} d_{ip}\lambda_p \geq 1, \qquad \forall i \in \mathcal{N}, \tag{7.7b}$$

$$\lambda_p \geq 0, \qquad \forall p \in \mathcal{P}'_{\mathcal{S},\mathcal{D}}. \tag{7.7c}$$

After solving the RMLP, let π_i be the dual variable associated with the i-th constraint (7.7b). The reduced cost for a column $p \in \mathcal{P}_{\mathcal{S},\mathcal{D}}$ is $r_p := 1 - \sum_{i\in\mathcal{N}} \pi_i d_{ip}$. If there is a column $p \in \mathcal{P}_{\mathcal{S},\mathcal{D}} \setminus \mathcal{P}'_{\mathcal{S},\mathcal{D}}$ whose reduced cost r_p is negative, then adding p to $\mathcal{P}'_{\mathcal{S},\mathcal{D}}$ could reduce the objective value of the RMLP. Otherwise, the solution for the RMLP is also optimal for the master LP problem. The column with the most negative reduced cost is determined by solving a pricing problem.

Before the column generation procedure is applied to the current node, the items that can only be packed together are combined into the set $\mathcal{S}$ using a preprocessing process. Let the new item set be $\mathcal{N}'$, a', b' be the merged parameters, and the new conflict relation be $\mathcal{D}'$. Preprocessing leads to a smaller pricing problem, which can be formulated to a *submodular knapsack problem with conflicts*:

$$\max \quad \sum_{i\in\mathcal{N}'} \pi'_i x_i, \tag{7.8a}$$

$$s.t. \quad \sum_{i\in\mathcal{N}'} a'_i x_i + \sigma\sqrt{\sum_{i\in\mathcal{N}'} b'_i x_i} \leq c, \tag{7.8b}$$

$$x_{i_1} + x_{i_2} \leq 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \tag{7.8c}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in \mathcal{N}'. \tag{7.8d}$$

If the optimal value $\sum_{i\in\mathcal{N}'} \pi'_i x_i > 1$, then the corresponding column has a negative reduced cost $1 - \sum_{i\in\mathcal{N}'} \pi'_i x_i$ and is added to the RMLP. Otherwise, the solution of the RMLP is optimal for the master LP, and the current node is solved. The details of the pricing algorithms can be found in Sect. 7.4.

Since, within a time limit, pricing problems may not be solved optimally, a pricing algorithm may find an existing column in $\mathcal{P}'_{\mathcal{S},\mathcal{D}}$ or a column with a positive reduced cost. Therefore, adding the column does not improve the RMLP, and the column generation procedure halts. The following simple constraint can exclude existing solutions from the pricing problem and thus shrink the search space:

$$\sum_{i\in\mathcal{N}'} \pi'_i x_i \geq 1 + \epsilon, \tag{7.9}$$

where ϵ is a sufficiently small positive real number. Exact algorithms can easily add this constraint to exclude existing columns in $\mathcal{P}'_{\mathcal{S},\mathcal{D}}$. This constraint also guarantees that if solutions of negative reduced costs exist, then exact algorithms can find one of them.

7.3.3 Primal heuristics

We discuss primal heuristics that help find primal feasible solutions to the set covering formulation. We use two heuristics: the first heuristic employs an approximation algorithm to find a primal solution that forms a set $\mathcal{P}'$ of initial columns, and the second heuristic tries to find a primal solution once a column is generated and added to $\mathcal{P}'$.

[97] propose approximation algorithms to find a feasible solution with $8/3$-ratio to the optimal solution to the submodular bin packing. Their algorithms are greedy and easy to implement, so we employ these algorithms as the first heuristic.

During column generation, each generated column could be combined with the previous columns in $\mathcal{P}'$ into a primal feasible solution. Our second primal heuristic is similar to the greedy column selection heuristic in [214, 185]. Once a column is generated, we force it into a potential solution. Then, we greedily select an existing column from $\mathcal{P}'$ that packs the maximum number of unpacked items until all items are packed. We note that the heuristic may find columns that do not improve the RMLP.

7.3.4 Dual bound computation

For an optimization problem, a dual bound certifies the optimality of a solution. In the branch-and-price setting, a local dual bound at each node of the search tree is a lower bound on the optimum of the master problem (7.6). The algorithm uses the local dual bound to fathom the node or select branch nodes.

The optimum of the master LP problem is a local dual bound. However, the column generation procedure usually needs to solve many pricing problems to converge to this optimum. At each iteration of the column generation procedure, another local dual bound is available. This bound is referred to in the literature as *Farley bound*. The following lemma illustrates how this bound can be computed.

Lemma 7.3 ([136, 294]). *Let v_{MP} be the optimum of the master LP, let v_{RMLP} be the optimum of the RMLP, let v_{price} be a dual bound for the pricing problem* (7.8)*, and let $v_{\mathrm{F}} := \frac{v_{\mathrm{RMLP}}}{v_{\mathrm{price}}}$ be the Farley bound. Then, $v_{\mathrm{F}} \leq v_{\mathrm{MP}}$, and thus v_{F} is a local dual bound.*

The computation of the Farley bound requires a dual bound on the pricing problem, obtained using an exact pricing algorithm. The branch-and-price algorithm holds a local lower bound v_{ld} at each search tree node. After solving each pricing problem, the branch-and-price algorithm updates v_{ld} according to the following rule:

$$v_{\mathrm{ld}} = \max\{v_{\mathrm{F}}, v_{\mathrm{ld}}\}.$$

Early stopping rules from [304] can compare the local dual bound and the primal bound to improve the branch-and-price algorithm. The rules exploit integrality and can stop column generation earlier than the classical algorithm. We implement these rules in our branch-and-price solver.

7.4 Solving the pricing problem

In this section, we present solution methods for the pricing problem. The proposed algorithms can be implemented as a stand-alone solver for the submodular knapsack problem.

We first present a fast pricing heuristic. We then present two formulations of the submodular knapsack problem (with conflicts): a convex BSOCP formulation and a non-convex MBQCP formulation. The convex BSOCP formulation is solved in our experiments for a comparative study. The PWL method is a way to approximate nonlinear functions (or relax under some conditions) by linear functions in its subdomain. We derive a PWL relaxation of the MBQCP formulation and develop an exact PWL-based branch-and-cut algorithm (PWL-B&C) for the pricing problem.

To speed up column generation, we also present a hybrid pricing strategy that can replace the exact pricing algorithm with a fast pricing heuristic.

7.4.1 Pricing heuristic

We propose a fast heuristic, the fixing-greedy heuristic. This heuristic is used by the hybrid pricing strategy to speed up the column generation procedure.

The fixing-greedy heuristic is based on the best-fit-greedy algorithm. The best-fit-greedy algorithm adds an item per iteration only if it does not conflict with the previously added items, as long as the capacity is not exceeded. The heuristic keeps

- Δ: the set of items added to the bin, which is initially empty.

At each iteration, the best-fit greedy heuristic has the following steps:

1. computes the sum of a'_i and the sum of b'_i of added items, i.e., $A := \sum_{i\in\Delta} a'_i$ and $B := \sum_{i\in\Delta} b'_i$;
2. find the set $\overline{\Delta} := \{i \in \mathcal{N}' \setminus \Delta : A + a'_i + \sigma\sqrt{B + b'_i} \leq c\}$ of items that can be added to the bin;
3. if $\overline{\Delta} = \emptyset$, exits and outputs Δ;
4. for each unadded item $i \in \overline{\Delta}$, computes the incremental capacity usage $\gamma_i := (A + a'_i + \sigma\sqrt{B + b'_i}) - (A + \sigma\sqrt{B})$, and the profit-over-usage ratio $r_i := \frac{\pi'_i}{\gamma_i}$;
5. adds the unadded item with the maximum r_i into Δ.

The fixing-greedy heuristic enforces, for each time, an item in $\mathcal{N}'$ to be in the solution, runs the best-fit greedy algorithm, and outputs the best solution.

7.4.2 BSOCP formulation

The Binary Second-Order Conic Programming formulation of the pricing problem (7.8) is similar to the BSOCP formulation of SMBP (7.4).

Applying the same technique in Remark 7.1, the BSOCP formulation of the pricing problem is:

$$\max \quad \sum_{i\in\mathcal{N}'} \pi'_i x_i, \tag{7.10a}$$

$$s.t. \quad \sum_{i\in\mathcal{N}'} a'_i x_i + \sigma\sqrt{\sum_{i\in\mathcal{N}'} b'_i x_i^2} \le c, \tag{7.10b}$$

$$x_{i_1} + x_{i_2} \le 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \tag{7.10c}$$

$$x_i \in \{0,1\}, \qquad \forall i \in \mathcal{N}'. \tag{7.10d}$$

Where (7.10b) can be represented by 3d second-order conic constraints. The BSOCP formulation (7.10) is a convex MINLP formulation.

In this section, we analyze the polyhedral outer approximation of the BSOCP formulation (7.10) and show that a finite number of cutting planes is sufficient to define an exact MILP reformulation of the BSOCP formulation (7.10).

To simplify the presentation, we use the following notation:

- the left-hand side of (7.10b):

$$f(x) := \sum_{i\in\mathcal{N}'} a'_i x_i + \sigma\sqrt{\sum_{i\in\mathcal{N}'} b'_i x_i^2};$$

- the binary set defined by (7.10b):

$$\mathcal{C} := \{x \in \{0,1\}^{\mathcal{N}'} : f(x) \le c\};$$

- the continuous relaxation of $\mathcal{C}$:

$$\overline{\mathcal{C}} := \{x \in [0,1]^{\mathcal{N}'} : f(x) \le c\}.$$

Since f is convex, $\overline{\mathcal{C}}$ is convex. We also note that the convex hull of $\mathcal{C}$ is a polytope. A set $\mathcal{O}$ is a polyhedral outer approximation of $\mathcal{C}$, if $\mathcal{O}$ is a polyhedron and $\mathcal{C} \subset \mathcal{O}$. A polyhedral outer approximation can be constructed as follows. Define a linearization of f at some $\hat{x}$ in the domain of f by $\mathcal{L}^f_{\hat{x}}(x) := f(\hat{x}) + \nabla f(\hat{x})^\top (x - \hat{x})$. Since f is convex, $\mathcal{L}^f_{\hat{x}}$ is an under-estimator of f, i.e., $\mathcal{L}^f_{\hat{x}}(x) \le f(x)$ for any x. Hence, $\mathcal{L}^f_{\hat{x}}(x) \le c$ is a linear inequality valid for $f(x) \le c$.

A polyhedral outer approximation $\mathcal{O}$ is said *exact*, if $\mathcal{O} \cap \{0,1\}^n = \mathcal{C}$. So, solving the optimization problem over an exact polyhedral outer approximation with binary and conflict constraints is equivalent to solving the submodular knapsack problem with conflicts. Next, we identify a family of valid inequalities that give an exact polyhedral outer approximation. Each of these valid inequalities corresponds to a binary point not in $\mathcal{C}$.

Theorem 7.4. *Given a point $\hat{x} \in \{0,1\}^{\mathcal{N}'}$, the following inequality is valid for $\mathcal{C}$ and $\overline{\mathcal{C}}$:*

$$\sum_{i\in\mathcal{N}'} a'_i x_i + \frac{\sigma}{\sqrt{\sum_{i\in\mathcal{N}'} b'_i \hat{x}_i}} \sum_{i\in\mathcal{N}'} b'_i \hat{x}_i x_i \le c. \tag{7.11}$$

Let

$$\mathcal{O} = \{x \in [0,1]^{\mathcal{N}'} : \sum_{i \in \mathcal{N}'} a'_i x_i + \frac{\sigma}{\sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i}} \sum_{i \in \mathcal{N}'} b'_i \hat{x}_i x_i \leq c,\ \forall \hat{x} \in \{0,1\}^{\mathcal{N}'} \setminus \mathcal{C}\}.$$

Moreover,

1. *if $\hat{x} \notin \mathcal{C}$, the valid inequality is violated by $\hat{x}$;*
2. *$\mathcal{O}$ is exact, and $\mathcal{C} = \mathcal{O} \cap \{0,1\}^{\mathcal{N}'}$.*

Proof. Since function f is convex, it follows that

$$\mathcal{L}^f_{\hat{x}}(x) \leq f(x) \leq c.$$

Moreover,

$$\begin{aligned}
&\mathcal{L}^f_{\hat{x}}(x) \\
=&f(\hat{x}) + \nabla f(\hat{x})^\top (x - \hat{x}) \\
=&\sum_{i \in \mathcal{N}'} a'_i x_i + \sigma \sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i^2} + \frac{\sigma}{\sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i^2}} \sum_{i \in \mathcal{N}'} b'_i \hat{x}_i (x_i - \hat{x}_i) \\
=&\sum_{i \in \mathcal{N}'} a'_i x_i + \sigma \sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i^2} + \frac{\sigma}{\sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i^2}} \sum_{i \in \mathcal{N}'} b'_i \hat{x}_i x_i - \frac{\sigma}{\sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i^2}} \sum_{i \in \mathcal{N}'} b'_i \hat{x}_i \hat{x}_i \\
=&\sum_{i \in \mathcal{N}'} a'_i x_i + \frac{\sigma}{\sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i}} \sum_{i \in \mathcal{N}'} b'_i \hat{x}_i x_i
\end{aligned}$$

where the last equation follows from the fact that $\hat{x}$ is binary.

Therefore, inequality (7.11) in the statement is valid for $\mathcal{C}$. The left-hand side of inequality (7.11) evaluated at $\hat{x}$ is $\sum_{i \in \mathcal{N}'} a'_i \hat{x}_i + \sigma \sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i}$ which is by hypothesis is at least c, so $\hat{x}$ violates the inequality.

Let us consider $x^* \in \{0,1\}^{\mathcal{N}'}$. If $x^* \notin \mathcal{C}$, then x^* violates the $\mathcal{L}^f_{x^*}(x) \leq c$ which is a facet defining inequality of $\mathcal{O}$, then $x^* \notin \mathcal{O}$. Hence, $x^* \in \mathcal{O}$ implies that $x^* \in \mathcal{C}$. If $x^* \in \mathcal{C}$, since $\mathcal{O}$ is a polyhedral outer approximation of $\mathcal{C}$, x^* must be in $\mathcal{O}$. Therefore, $\mathcal{C} = \mathcal{O} \cap \{0,1\}^{\mathcal{N}'}$. □

Looking at the above theorem, we find that each binary point not in $\mathcal{C}$ gives rise to a valid inequality separating it from $\mathcal{C}$. Moreover, binary points in $\mathcal{C}$ satisfy these valid inequalities, i.e., they are in the polyhedral outer approximation $\mathcal{O}$. We define two sets related to the polyhedral outer approximation $\mathcal{O}$. The *generating set* is defined as

$$\mathcal{X} := \{\hat{x} \in \{0,1\}^{\mathcal{N}'} : \hat{x} \notin \mathcal{C}\}, \tag{7.12}$$

because it generates the following *cut coefficient set*:

$$\Theta := \left\{ \theta \in \mathbb{R}^{\mathcal{N}'} : \exists \hat{x} \in \mathcal{X}\ \forall i \in \mathcal{N}'\ \theta_i = a'_i + \frac{\sigma}{\sqrt{\sum_{i \in \mathcal{N}'} b'_i \hat{x}_i}} b'_i \hat{x}_i \right\}. \tag{7.13}$$

By Thm. 7.4, $\mathcal{O}$ is an exact polyhedral outer approximation, so replacing $x \in \mathcal{C}$ with $x \in \mathcal{O}$ does not change the binary feasible set. This gives rise to an exact MILP formulation equivalent

to the submodular knapsack problem with conflicts:

$$\max \quad \sum_{i\in\mathcal{N}'} \pi'_i x_i, \tag{7.14a}$$

$$s.t. \quad \theta^\top x \le c, \qquad \forall \theta \in \Theta \tag{7.14b}$$

$$x_{i_1} + x_{i_2} \le 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \tag{7.14c}$$

$$x_i \in \{0,1\}, \qquad \forall i \in \mathcal{N}'. \tag{7.14d}$$

However, $\mathcal{X}$ (and hence Θ) is unknown before exploring the search space, and its cardinality may be exponential. In practice, the cuts corresponding to Θ can only be separated *lazily*, i.e., a cut is added until a point $\hat{x}$ is found in $\mathcal{X}$. Off-the-shelf solvers do not use this finite family of cuts, but it is a crucial component for constructing our PWL-B&C algorithm in Sect. 7.4.5.

The following lemma explains the approximation error of the polyhedral outer approximation $\mathcal{O}$ w.r.t. $\overline{\mathcal{C}}$.

Lemma 7.5 ([51])**.** *Let $\epsilon > 0$, then there exists a method to construct a polyhedral outer approximation $\mathcal{O}$ of $\overline{\mathcal{C}}$ with additional $\mathcal{O}(1)|\mathcal{N}'|\log(\frac{1}{\epsilon})$ variables and constraints, such that the relative ℓ_∞ approximation error $\max_{x\in\mathcal{O}} |\sum_{i\in\mathcal{N}'} a'_i x_i + \sigma\sqrt{\sum_{i\in\mathcal{N}'} b'_i x_i^2} - c|/c$ is at most ϵ.*

Note that the approximation error of the polyhedral outer approximation depends on the number of variables.

7.4.3 MBQCP formulation

We present a non-convex Mixed Binary Quadratically Constrained Programming formulation for the submodular knapsack problem (with conflicts). Although we do not use this formulation to solve the pricing subproblems, this formulation inspires PWL relaxation and the PWL-B&C algorithm. Here, we introduce a slack variable w to define the sum $\sum_{i\in\mathcal{N}'} a'_i x_i$. Then our MBQCP formulation becomes the following non-convex MINLP program:

$$\max \quad \sum_{i\in\mathcal{N}'} \pi'_i x_i, \tag{7.15a}$$

$$s.t. \quad \sum_{i\in\mathcal{N}'} a'_i x_i = w, \tag{7.15b}$$

$$\sigma^2 \sum_{i\in\mathcal{N}'} b'_i x_i \le (c-w)^2, \tag{7.15c}$$

$$x_{i_1} + x_{i_2} \le 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \tag{7.15d}$$

$$x_i \in \{0,1\}, \qquad \forall i \in \mathcal{N}', \tag{7.15e}$$

$$w \in [0, c]. \tag{7.15f}$$

Although the program contains a concave quadratic constraint (7.15c), the nonlinearity is only a univariate quadratic function compared to the $|\mathcal{N}'|$-dimensional nonlinear SOC function f in (7.10b).

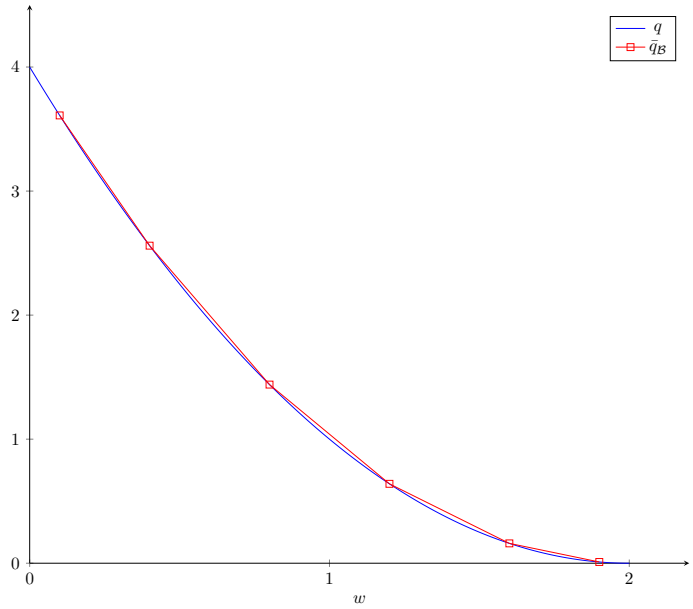


Figure 7.1 Graphs of the quadratic and its PWL over-estimator

7.4.4 PWL relaxation

A Piece-Wise Linear (PWL) function is linear on each piece of a given partition of its domain. We derive a MILP relaxation of the MBQCP formulation (7.15) based on the PWL relaxation for the quadratic function, and refer to this new MILP relaxation as the PWL relaxation. The approximation error of the optimal PWL relaxation is discussed in this section. Let us denote by $q(w) := (c - w)^2$ the univariate quadratic function. We denote a value of the slack variable w in the constraint (7.15c) as a *breakpoint*. Given an ordered set of breakpoints $\mathcal{B} = (w_1, w_2, \ldots, w_h)$ such that $w_k \in [\underline{w}, \overline{w}]$ ($k \in [h] := \{1, \cdots, h\}$), $w_1 = \underline{w}$ and $w_h = \overline{w}$, the following function is a PWL approximation of q over the domain $[\underline{w}, \overline{w}]$:

$$\bar{q}_{\mathcal{B}}(w) := \frac{q(w_k) - q(w_{k-1})}{w_k - w_{k-1}}(w - w_{k-1}) + q(w_{k-1}), \text{ for } w_{k-1} \leq w \leq w_k, 2 \leq k \leq h.$$

Note that $\bar{q}_{\mathcal{B}}$ is an *over-estimator* of q due to the convexity of q.

We call $\mathcal{B}$ a breakpoint set in $[\underline{w}, \overline{w}]$, and $\bar{q}_{\mathcal{B}}$ its induced PWL function. Note that we consider the two bounds $\underline{w}$ and $\overline{w}$ as breakpoints here. Fig. 7.1 shows the graphs of a quadratic function and its PWL over-estimator, where $\underline{w} = 0.1$, $\overline{w} = 1.9$, $c = 2$, and $\mathcal{B} = \{0.1, 0.4, 0.8, 1.2, 1.6, 1.9\}$.

Assume that we are given the breakpoints $\mathcal{B}$. Replacing $\sigma^2 \sum_{i \in \mathcal{N}'} b'_i x_i \leq q(w)$ with $\sigma^2 \sum_{i \in \mathcal{N}'} b'_i x_i \leq \bar{q}_{\mathcal{B}}(w)$ in the constraint (7.15c), we obtain the following PWL relaxation of the MBQCP formulation (7.15):

$$\max \quad \sum_{i\in\mathcal{N}'} \pi_i' x_i, \tag{7.16a}$$

$$s.t. \quad \sum_{i\in\mathcal{N}'} a_i' x_i = w, \tag{7.16b}$$

$$\sigma^2 \sum_{i\in\mathcal{N}'} b_i' x_i \le \bar{q}_{\mathcal{B}}(w), \tag{7.16c}$$

$$x_{i_1} + x_{i_2} \le 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \tag{7.16d}$$

$$x_i \in \{0,1\}, \qquad \forall i \in \mathcal{N}', \tag{7.16e}$$

$$w \in [0, c]. \tag{7.16f}$$

Remark 7.6. *(Modeling PWL functions) The graphs of PWL functions have several MILP formulations, see [299]. In this chapter, we consider the logarithmic model. We denote by z the auxiliary binary variables introduced in the MILP formulation of $\bar{q}_{\mathcal{B}}$. From version 20.1.0 [180], CPLEX can automatically formulate $\bar{q}_{\mathcal{B}}$ to the logarithmic model and add auxiliary variables z in the internal data structure.*

The approximation error of a PWL relaxation is expressed as ℓ_p-norm of the difference between the approximation function and the target function.

Definition 7.7. *Given a set $\mathcal{B} \subset [\underline{w}, \overline{w}]$ of breakpoints, the ℓ_p approximation error of $\bar{q}_{\mathcal{B}}$ with respect to q over $[\underline{w}, \overline{w}]$ is defined as $\ell_p(\bar{q}_{\mathcal{B}}, q) := (\int_{\underline{w}}^{\overline{w}} |\bar{q}_{\mathcal{B}}(w) - q(w)|^p \, dw)^{\frac{1}{p}}$.*

Since the approximation error measures the quality of a PWL approximation to the quadratic function, thus it in turn measures the error of the PWL relaxation to the MBQCP formulation. Empirically, the optimal solution to a PWL relaxation with a small approximation error should have a small gap to the optimal solution of the submodular knapsack with conflicts. On the other hand, although adding breakpoints decreases the approximation error, it increases the computation resource to solve the PWL relaxation. So a common problem is understanding the best possible approximation error given a fixed number of breakpoints (limited computational resource).

This problem can be formalized as follows. Given an integer h (number of breakpoints), denote by $\mathbb{B}^h$ the family of breakpoint sets of cardinality h in $[\underline{w}, \overline{w}]$, the *breakpoint selection problem* aims to find a set $\mathcal{B} \in \mathbb{B}^h$ to minimize the ℓ_p error:

$$\min_{\mathcal{B}\in\mathbb{B}^h} \ell_p(\bar{q}_{\mathcal{B}}, q). \tag{7.17}$$

A convex program [149] can compute the ℓ_∞-approximation error for general nonconvex functions. An error analysis [54] gives asymptotically tight bounds to quantify the ℓ_2-approximation error.

The following theorem gives the best ℓ_∞-approximation error that we can achieve: an optimal solution to the breakpoint selection problem under the ℓ_∞-approximation error is an equidistant partition of $[\underline{w}, \overline{w}]$.

Theorem 7.8. *Given $\mathcal{B} \in \mathbb{B}^h$,*

$$\ell_\infty(\bar{q}_{\mathcal{B}}, q) = \max_{w\in[\underline{w},\overline{w}]} |\bar{q}_{\mathcal{B}}(w) - q(w)| = \max_{2\le k\le h} \frac{(w_k - w_{k-1})^2}{4}.$$

Furthermore, let $w_k = \underline{w} + \frac{k-1}{h-1}(\overline{w} - \underline{w})$ *for* $1 \leq k \leq h$*, which yields the minimum* ℓ_∞*-approximation error* $\frac{(\overline{w}-\underline{w})^2}{4(h-1)^2}$ *for the break point selection problem* (7.17)*.*

Proof. Since $\bar{q}_{\mathcal{B}}$ and q have the same value at $w \in \{w_1, \ldots, w_h\}$, it follows that the ℓ_∞-norm is the maximum value of ℓ_∞-norms over individual sub intervals:

$$\ell_\infty(\bar{q}_{\mathcal{B}}, q) = \max_{w\in[\underline{w},\overline{w}]} |\bar{q}_{\mathcal{B}}(w) - q(w)| = \max_{2\leq k\leq h} \max_{w\in[w_{k-1},w_k]} |\bar{q}_{\mathcal{B}}(w) - q(w)|.$$

Let $w \in [w_{k-1}, w_k]$, then

$$\begin{aligned}
&|\bar{q}_{\mathcal{B}}(w) - q(w)| \\
=&\frac{q(w_k) - q(w_{k-1})}{w_k - w_{k-1}}(w - w_{k-1}) + q(w_{k-1}) - (c - w)^2 \\
=&(w - w_{k-1})(w_k - w).
\end{aligned}$$

We have

$$\begin{aligned}
&\max_{w\in[w_{k-1},w_k]} |\bar{q}_{\mathcal{B}}(w) - q(w)| \\
=&\max_{w\in[w_{k-1},w_k]} (w - w_{k-1})(w_k - w) \\
=&\frac{(w_k - w_{k-1})^2}{4}.
\end{aligned}$$

The maximum value is at $w = \frac{w_{k-1}+w_k}{2}$.

It follows that (7.17) is equivalent to:

$$\min_{\underline{w}=w_1\leq\ldots\leq w_h=\overline{w}} \max_{2\leq k\leq h} \frac{(w_k - w_{k-1})^2}{4}.$$

Therefore, the optimal solution is an equidistant partition of $[\underline{w}, \overline{w}]$, and the results follow. □

The approximation error decreases with the quadratic rate with respect to h. The relative ℓ_∞-approximation error is defined as

$$\frac{\ell_\infty(\bar{q}_{\mathcal{B}}, q)}{(\overline{w} - \underline{w})^2}.$$

We have the following result on the relative approximation error of the PWL relaxation.

Corollary 7.9. *Let* $\epsilon > 0$*, then there exists a MILP formulation of PWL function* $\bar{q}_{\mathcal{B}}$ *induced by* $\mathcal{B}$ *with* $\mathcal{O}(1)\log(\frac{1}{\epsilon})$ *binary variables and* $\mathcal{O}(1)\frac{1}{\sqrt{\epsilon}}$ *continuous variables and constraints, such that the relative* ℓ_∞*-approximation error is at most* ϵ*.*

Proof. For the logarithmic model of PWL function, given h breakpoints from the equidistant partition, the relative ℓ_∞-approximation error is $\frac{(\overline{w}-\underline{w})^2}{4(h-1)^2(\overline{w}-\underline{w})^2} = \frac{1}{4(h-1)^2}$ with $\log(h-1)$ binary variables and $h-1$ continuous variables and constraints [299], the result follows. □

Next, we summarize the approximation errors of two relaxations to their corresponding formulations. Note that we do not consider the integrality of the binary variable x'. Comparing Lemma 7.5 and Cor. 7.9, the approximation error of the PWL relaxation (7.16) to the MBQCP

formulation (7.15) is independent of the number of variables, while the approximation error of the polyhedral outer approximation to the BSOCP formulation (7.10) depends on this number.

We remark that our PWL relaxation differs from [279]'s PWL relaxation. The constraint (7.15) of MBQCP formulation is equivalent to $\sigma\sqrt{\sum_{i\in\mathcal{N}'} b_i' x_i} \leq c - w$, and a PWL relaxation was used for the left-hand side concave function $\sigma\sqrt{\sum_{i\in\mathcal{N}'} b_i' x_i}$ in [279]. However, the optimal approximation error for such PWL relaxation has yet to be discovered.

7.4.5 Exact PWL-B&C algorithm

The approximation error of the PWL relaxation is dimensionless but only for a small number of breakpoints, it is not exact. Instead of adding many breakpoints, the finite number of cuts induced by the set Θ in (7.13) suffices to make the PWL relaxation exact. To solve it, we propose a combined formulation and a branch-and-cut algorithm based on the PWL relaxation (PWL-B&C).

$$\max \quad \sum_{i\in\mathcal{N}'} \pi_i' x_i, \tag{7.18a}$$

$$s.t. \quad \sum_{i\in\mathcal{N}'} a_i' x_i = w, \tag{7.18b}$$

$$\sigma^2 \sum_{i\in\mathcal{N}'} b_i' x_i \leq \bar{q}_{\mathcal{B}}(w), \tag{7.18c}$$

$$\theta^\top x \leq c, \qquad \forall \theta \in \Theta \tag{7.18d}$$

$$x_{i_1} + x_{i_2} \leq 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \tag{7.18e}$$

$$x_i \in \{0,1\}, \qquad \forall i \in \mathcal{N}', \tag{7.18f}$$

$$w \in [0, c]. \tag{7.18g}$$

Formulation (7.18) combines the MILP formulation (7.14) with the (redundant) PWL relaxation. As already mentioned by Thm. 7.4, the MILP formulation (7.14) is an exact formulation for submodular knapsack problems with conflicts, so this combined formulation (7.18) is also exact.

The intuition underlying the combined formulation (7.18) is that we cannot add numerous valid inequalities (7.18d) *a priori*. In practice, we add them *lazily* to exclude infeasible binary solutions to the submodular knapsack with conflicts in the course of the search, and this method is typically supported or suggested by *lazy cut callbacks* of some solvers such as `CPLEX` and `SCIP`. However, in this way, we cannot control the initial relaxation quality given solely by a few valid inequalities from (7.18d). On the contrary, the PWL relaxation (7.18c) can be enforced *a priori*, and its quality is controllable (Thm. 7.8). So we can leverage it to reduce the initial search space and refine the relaxation by adding valid inequalities lazily. This intuition and formulation give rise to a tailored Algorithm 7.1 for submodular knapsack with conflicts, partly inspired by algorithms in [96]. We show in experiments that this formulation with redundant constraints (7.18b) and (7.18c) can be solved much faster than the standard BSOCP formulation (7.10). In practice, only a few cuts in (7.18d) must separate before the convergence.

Our algorithm consists of three main steps: tightening the bounds, constructing the PWL relaxation (breakpoints), and the PWL B&C algorithm. First, bound tightening is a preprocess-

ing procedure used to tighten the bounds on the breakpoints for all pricing problems. Then, the PWL relaxation (breakpoints) is constructed for all pricing problems, and this is also a pre-solving procedure. The construction depends on the number of items, the size of the items, and the capacity. Finally, based on the PWL relaxation, the PWL-B&C algorithm is adapted to the LP-B&C algorithm [96].

Bound tightening The bound tightening procedure is called before the branch-and-price algorithm to shrink the boundaries of the breakpoints $\mathcal{B}$ into $[0, c]$.

Considering a pricing problem at a node of the search tree, we find that if $w = \sum_{i\in\mathcal{N}'} a'_i x_i$ is small, $q(w) = (c - w)^2$ is larger than $\sigma^2 \sum_{i\in\mathcal{N}'} b'_i x_i$, so the capacity constraint (7.15c) is not active. Thus, there is no need to overestimate q when w is small. More precisely, there is a $\underline{w} \in [0, c]$ such that, for any binary solution $x \in \{0, 1\}^{\mathcal{N}'}$, let $w = \sum_{i\in\mathcal{N}'} a'_i x_i$, if $w \le \underline{w}$, then $\sigma^2 \sum_{i\in\mathcal{N}'} b'_i x_i \le q(\underline{w})$. Since q is non-increasing, $q(w) \ge q(\underline{w}) \ge \sigma^2 \sum_{i\in\mathcal{N}'} b'_i x_i$. The point $\underline{w}$ is called *lower breakpoint*, the submodular capacity constraint (7.15c) is never violated for $w \in [0, \underline{w}]$. We can start by overestimating q starting from the maximum lower breakpoint computed from the following convex MBQCP problem:

$$
\begin{aligned}
\underline{w} := \max \quad & w, \\
s.t. \quad & \sum_{i\in\mathcal{N}'} a'_i x_i = w, \\
\sigma^2 & \sum_{i\in\mathcal{N}'} b'_i x_i \ge (c - w)^2, \\
& x_{i_1} + x_{i_2} \le 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \\
& x_i \in \{0, 1\}, \qquad \forall i \in \mathcal{N}'.
\end{aligned}
\tag{7.19}
$$

Similarly, we can define the *upper breakpoint*. There exists some upper breakpoint $\overline{w} \in [0, c]$, such that, for every binary solution $x \in \{0, 1\}^{\mathcal{N}'}$, if $\sum_{i\in\mathcal{N}'} a'_i x_i + \sigma\sqrt{\sum_{i\in\mathcal{N}'} b'_i x_i^2} \le c$, then $\sum_{i\in\mathcal{N}'} a'_i x_i \le \overline{w}$. The minimum upper breakpoint can be computed from the following BSOCP problem:

$$
\begin{aligned}
\overline{w} := \max \quad & \sum_{i\in\mathcal{N}'} a'_i x_i, \\
s.t. \quad & \sum_{i\in\mathcal{N}'} a'_i x_i + \sigma\sqrt{\sum_{i\in\mathcal{N}'} b'_i x_i^2} \le c, \\
& x_{i_1} + x_{i_2} \le 1, \qquad \forall (i_1, i_2) \in \mathcal{D}', \\
& x_i \in \{0, 1\}, \qquad \forall i \in \mathcal{N}'.
\end{aligned}
\tag{7.20}
$$

We solve the above two programs at the root node and obtain the bound $[\underline{w}_r, \overline{w}_r]$ for breakpoints. Since the feasible sets of the other nodes are a subset of the root node set, the above programs at other nodes are more strict than those at the root node. It follows for $\underline{w}, \overline{w}$ of any other node that $\underline{w}_r \le \underline{w}$ and $\overline{w} \le \overline{w}_r$. We then set $\underline{w} = \underline{w}_r$ and $\overline{w} = \overline{w}_r$ for all nodes.

Construction of breakpoints To determine the number of breakpoints $\mathcal{B}$, we run a greedy heuristic algorithm that tries to maximize the number of items in a bin. We take h as the

solution value given by the heuristic algorithm and assign breakpoints h equidistantly in $[\underline{w}, \overline{w}]$. The equidistant partition gives the best approximation error according to Thm. 7.8 for a fixed number of breakpoints. We also add a breakpoint corresponding to $w = 0$.

PWL-B&C algorithm The main steps of the PWL-B&C algorithm are described in Algorithm 7.1. Recall that the problem (7.8) is a maximization problem. Algorithm 7.1 maintains a set of active nodes $\mathscr{N}$ of the search tree, a pool of cuts $\mathscr{C}$, an incumbent solution x^* ($\sum_{i\in\mathcal{N}'} \pi_i' x_i^*$ is a primal bound).

A node (l, u, U) is characterized by the finite variable boundary vectors l and u and the node's dual upper bound U. The upper bound U is inherited from its parent node and computed via the LP relaxation. Note that the PWL function is a modeling concept. We use a MILP solver, i.e., `CPLEX`, that formulates the PWL function $q_{\mathcal{B}}$ into a MILP. We denote by z the additional binary variables to model $\bar{q}_{\mathcal{B}}$ (see Sect. 7.4.4). The variables z are also constructed internally by `CPLEX`, and we assume that the PWL function is forced when z is set to binary.

We denote by $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l, u, U)$ the MILP relaxation restricted to finite bounds (l, u) for (x, z) at a node of the search tree. The MILP relaxation $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l, u, U)$ consists of the PWL relaxation (7.16), cuts from $\mathscr{C}$, and other cuts added by the MILP solver.

Algorithm 7.1: PWL-B&C algorithm

Input: a submodular knapsack problem with conflicts (7.18), and the set $\mathcal{B}$ of breakpoints;
Output: a primal solution x^* and a dual upper bound (dual gap);
initialize MILP $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l_0, u_0, \infty)$ as the PWL relaxation (7.16) ; ▷ the PWL function is modeled by auxiliary binary variable z
initialize cut pool $\mathscr{C}$ to $\emptyset$, the node list $\mathscr{N}$ of $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l_0, u_0, \infty)$ with root node (l_0, u_0), incumbent solution $x^* = 0$, and the upper bound U of the root node to ∞;
while $\mathscr{N}$ *contains nodes* **do**
    remove a node (l, u) from $\mathscr{N}$;
    solve LP relaxation of $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l, u, U)$;
    if *LP is infeasible* **then**
        continue ; ▷ fathomed by infeasibility
    get an LP optimal solution $(\hat{x}, \hat{z})$;
    if *upper bound* $U \leq \sum_{i\in\mathcal{N}'} \pi_i \hat{x}_i$ **then**
        continue ; ▷ fathomed by bound
    else
        set U to $\sum_{i\in\mathcal{N}'} \pi_i \hat{x}_i$; ▷ update the dual upper bound
    end
    if $(\hat{x}, \hat{z})$ *is binary* **then**
        if $\hat{x}$ *satisfies capacity constraint* (7.8b) **then**
            set x^* to $\hat{x}$;
            continue ; ▷ fathomed by integrality
        else
            add separation cut to $\mathscr{C}$ by Thm. 7.4;
            add the node $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l, u, U)$ to $\mathscr{N}$;
            continue ; ▷ reoptimization after cut added
        end
    end
    add branch nodes to $\mathscr{N}$ using $(\hat{x}, \hat{z})$ (fractional) and U;
end

The node set $\mathscr{N}$ initially contains the root node (l^0, u^0), where $l^0, u^0 \in \mathbb{R}^I$ are the finite initial global bounds on variables (x, z). On Line 6 of Algorithm 7.1, the main loop removes a node (l, u, U) from $\mathscr{N}$. Line 7 solves the LP relaxation of $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l, u, U)$ by the MILP solver.

If the LP-relaxation $\mathscr{M}_{\mathcal{B}}(\mathscr{C}, l, u, U)$ is infeasible, Line 9 immediately fathoms the node by infeasibility. The upper bound U of the node means that any feasible solution to the combined formulation (7.18) that satisfies the bounds of the node for (x, z) has an objective value of at most U. Since LP is a relaxation of the combined formulation (7.18), any feasible solution to the combined formulation (7.18) that satisfies the bounds of the node for x has a objective value of at most U.

Line 12 fathoms the node by bound if U is not better than the incumbent value. Otherwise, the upper bound U of the node is set to the optimal value of LP on Line 14.

If $\hat{z}$ is not binary, then the PWL function is not implicitly enforced by the integrality of $\hat{z}$, so the algorithm should continue to branch. If $\hat{z}$ is binary (the PWL function is enforced) and $\hat{x}$ is binary, then the algorithm examines the solution $\hat{x}$.

If additionally, $\hat{x}$ is feasible (the capacity constraint is satisfied), then its objective value should be at least the upper bound U. Line 18 stores the new incumbent solution $\hat{x}$, and Line 19 fathoms the node since $\hat{x}$ is an optimal binary solution with respect to the bounds (l, u). Otherwise, the constraint (7.18d) is violated. Line 21 adds this constraint to the cut pool $\mathscr{C}$, Line 22 adds the current node for re-optimization, and Line 23 discards $\hat{x}$ by the cut in the next optimization iteration. Finally, $(\hat{x}, \hat{z})$ must be fractional on Line 26, the algorithm branches using the information from fractionality and U.

We remark that the idea in [279] for exact algorithms does not deploy cutting planes for approximating submodular knapsack, so in each iteration, new breakpoints are added to PWL, relaxations, and the underlying MILP solver needs restarts.

7.4.6 Hybrid pricing strategy

The pricing heuristic in Sect. 7.4.1 is fast, but it cannot guarantee the dual upper bound required by the Farley bound of Lemma 7.3. The exact pricing algorithm is slow but yields the dual upper bound for the pricing problem. The hybrid pricing strategy first calls the pricing heuristic to decide whether the exact pricing algorithm can improve the local dual bound of the master problem.

In fact, the exact algorithm is required only under a particular condition. The following proposition gives the condition.

Proposition 7.10. *Let v_{heur} be the solution value of the pricing heuristic, let v_{RMLP} be the optimum of RMLP* (7.7)*, and let v_{ld} be the current local dual bound for the master problem. If $\frac{v_{\mathrm{RMLP}}}{v_{heur}} \leq v_{\mathrm{ld}}$, the exact algorithm cannot yield a better local dual bound than v_{ld}.*

Proof. Let v_{popt} be the optimum for the pricing problem (7.8), then $v_{heur} \leq v_{popt}$. It follows that $\frac{v_{\mathrm{RMLP}}}{v_{popt}} \leq \frac{v_{\mathrm{RMLP}}}{v_{heur}} \leq v_{\mathrm{ld}}$. However, v_{popt} is the smallest pricing dual bound v_{price}, so $\frac{v_{\mathrm{RMLP}}}{v_{popt}}$ is the greatest Farley bound according to Lemma 7.3. Therefore even if the pricing algorithm is solved to optimality, we cannot obtain a better bound than v_{ld}. □

If the condition $\frac{v_{\mathrm{RMLP}}}{v_{heur}} \leq v_{\mathrm{ld}}$ holds, one can get rid of the exact pricing algorithm, and use the solution from the fixing-greedy heuristic in Sect. 7.4.1. The hybrid pricing strategy is outlined in Algorithm 7.2.

Algorithm 7.2: Hybrid pricing strategy

1 **Input:** a pricing problem (7.8) with the objective coefficients π', v_{RMLP} the optimum of RMLP (7.7), v_{ld} the local dual bound of the master problem;
2 **Output:** a generated column x^*, and the updated local dual bound v_{ld};
3 call the pricing heuristic with the objective coefficients π' ; ▷ run heuristic first
4 let x, v_{heur} be the heuristic solution and its value;
5 **if** $\frac{v_{\mathrm{RMLP}}}{v_{heur}} \leq v_{\mathrm{ld}}$ *and* $1 - \sum_{i \in \mathcal{N}'} \pi'_i \bar{x}_i < 0$ **then**
6 $x^* \leftarrow x$; ▷ heuristic solution
7 **else**
8 call the exact pricing Algorithm 7.1 ; ▷ exact pricing
9 let $\tilde{x}, v_{\mathrm{price}}$ be the primal solution and the dual bound;
10 $x^* \leftarrow \tilde{x}$;
11 $v_{\mathrm{ld}} = \max\{v_{\mathrm{ld}}, \frac{v_{\mathrm{RMLP}}}{v_{\mathrm{price}}}\}$; ▷ update the local dual bound
12 **end**

The heuristic algorithm is called first in Line 3. If $\frac{v_{\mathrm{RMLP}}}{v_{heur}} \leq v_{\mathrm{ld}}$, the exact pricing is not needed. If the heuristic solution x has a negative reduced cost, the strategy outputs it in Line 6. Otherwise, the strategy calls the exact algorithm in Line 8.

7.5 Computational experiments

In this section, we present the computational experiments we made to test the effectiveness of our branch-and-price algorithms for SMBP. In particular, we test different configurations of branch-and-price algorithms to evaluate the proposed techniques. The source code and benchmarks are publicly available on the project website https://github.com/lidingxu/cbp. We also provide a bash file to reproduce the experiments on Linux systems.

7.5.1 Benchmarks

We produce benchmarks as described in [97]. The authors test their approximation algorithms on benchmarks from real cloud data centers of `Google`, which are not accessible due to confidentiality [1].

They also describe data generation methods by considering a variety of uncertainty models, and these methods have a probabilistic interpretation: parameters of SMBP instances are derived from parameters of uncertainty models. For different risk levels α, they propose three data generation methods (cases) to construct the data a, b, σ in SMBP (7.4), i.e., the Gaussian case, the Hoeffding inequality case, and the distributionally robust approximation case.

We describe the generation methods next. In summary, we first determine overall parameters such as capacity and item numbers, then we generate distributions, and finally cast parameters of distributions into parameters of items.

The overall parameters of instances are set as follows. We set the capacity of each bin to 72 (the number of cores of the servers), the risk level $\alpha \in \{0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$. We set the number of items (i.e., jobs) $|\mathcal{N}| \in \{100, 400, 1000\}$ to obtain three benchmarks with different sizes: `CloudSmall`, `CloudMedium`, and `CloudLarge`. There are three generation methods and six risk levels.

[1] We, therefore, create new instances using the same generation method.

The distributions of instances are set as follows. We call the distribution of μ_i the *target distribution* for item i. We assume that every μ_i follows the same target distribution. This target distribution is unknown in [97] except for its quantiles in Table 7.1.

Given α and $\mathcal{N}$, we generate an SMBP instance as follows:

1. sample μ_i ($i \in \mathcal{N}$) according to Table 7.1;
2. sample a and b from μ and σ, using one of the following cases:
   - Gaussian case;
   - Hoeffding's inequality case;
   - distributionally robust approximation case.

Table 7.1 Example distribution of item size

Item sizes	1	2	4	8	16	32	72
% Items	36.3	13.8	21.3	23.1	3.5	1.9	0.1

We first illustrate the approach of sampling μ. We approximate the target distribution by a normalized histogram such that its quantile distribution is the same as in Table 7.1. A histogram consists of intervals divided from the entire range $[0, 72]$, and each interval has endpoints of two consecutive quantiles of Table 7.1. The histogram gives a discrete non-parametric estimation of the target distribution. We apply a two-stage sampling to obtain a nominal item size μ_i ($i \in \mathcal{N}$) sampled from a continuous distribution. It has two steps:

1. sample an interval $[d_1, d_2]$ from the histogram;
2. sample a nominal item size μ_i from $[d_1, d_2]$ uniformly.

Second, we construct a truncated Gaussian, which is defined by its lower and upper bounds $\underline{A}$ and $\overline{A}$, mean μ', and its standard deviation σ'. To obtain these parameters, for each $i \in \mathcal{N}$, we:

1. sample $\underline{A}_i \in [0.3, 0.6]$ and $\overline{A}_i \in [0.7, 1.0]$ uniformly;
2. sample scale parameter $s_i \in [0.1, 0.5]$;
3. compute the mean μ'_i and the standard variation σ'_i of the truncated Gaussian with lower bound $\underline{A}_i$, upper bound $\overline{A}_i$ and scale parameter s_i.

With the above parameters, we generate the data a, b, σ of SMBP (7.4). There are three cases, which correspond to different assumptions on the uncertainty or probability distribution.

For the Gaussian case:

1. let $\sigma = \Phi^{-1}(\alpha)$, where Φ is the cumulative distribution function of the Gaussian distribution;
2. for $i \in \mathcal{N}$, let $a_i = \mu'_i \mu_i$ and $b_i = (\sigma'_i \mu_i)^2$.

For the Hoeffding's inequality case:

1. let $\sigma = \sqrt{-0.5 \ln (1 - \alpha)}$;

2. for $i \in \mathcal{N}$, let $a_i = \mu_i' \mu_i$ and $b_i = ((\overline{A}_i - \underline{A}_i)\mu_i)^2$

For the distributionally robust approximation case:

1. let $\sigma = \sqrt{\alpha/(1-\alpha)}$;
2. for $i \in \mathcal{N}$, let $a_i = \mu_i' \mu_i$ and $b_i = (\sigma_i' \mu_i)^2$.

For all the above cases, if there exists $i \in \mathcal{N}$ such that a_i, b_i are too large to fit a bin (usually for large α, σ), then we rescale a_i, b_i to fit the bin.

We generate six instances with different random seeds for each combination of generation methods and risk levels. As a result, we have $108 = 6 \times 6 \times 3$ instances in a benchmark.

7.5.2 Experimental setups

In this section, we describe the setup of the experiments, including the development environment, the implementation of the algorithms, and the solution statistics.

Development environment The experiments are conducted on a server with Intel Xeon W-2245 CPU @ 3.90GHz, 126GB main memory, and Ubuntu 18.04 system. We use `SCIP` 8.0.1 [147] as a branch-and-price (B&P) framework to solve the set cover formulation (7.5). We use ILOG `CPLEX` 22.1 as:

- an LP solver to solve the RMLP (7.7);
- a BSOCP solver to solve the BSOCP formulations of SMBP (7.4) and the submodular knapsack problem with conflicts (7.10);
- a MILP solver used by the PWL-B&C Algorithm 7.1;

`CPLEX`'s parameters are set by default, except we disable its parallelism.

Solver implementation We implement four solvers for SMBP according to the proposed techniques in this chapter. Four of them are branch-and-price solvers. These solvers are as follows:

1. BSOCP-BC: a solver using `CPLEX`'s B&C algorithm to solve the compact BSOCP formulation of SMBP.
2. DW-BC: a B&P solver for solving the set cover formulation (7.5), which uses `CPLEX`'s B&C algorithm to solve the BSOCP formulation (7.10) of the pricing problem.
3. DW-PWL: a B&P solver for solving the set cover formulation (7.5), which uses the PWL-B&C algorithm to solve the combined formulation (7.16) of the pricing problem.
4. DW-Hybrid: DW-PWL enhanced with the hybrid pricing strategy in Algorithm 7.2.

We use the approximation algorithm from [97] to find an initial feasible solution that serves as a warm start for all solvers. All B&P solvers deploy the column selection heuristic in Sect. 7.3.3, and all exact pricing solvers add the solution exclusion constraint (7.9) to pricing problems. The time limit for each solver is 3600 CPU seconds.

If the column generation procedure at the root node does not finish after 3500 CPU seconds, it is halted, giving SCIP 100 CPU seconds to invoke its own primal heuristic.

For the pricing problems, we set the same time limit for the exact algorithms ($|\mathcal{N}| \times 0.015$ CPU seconds) and the same tolerance for relative gaps.

Performance metrics and statistical tests In order to evaluate the solver performance in different instances, we compute shifted geometric means (SGMs) (see [5]) of performance metrics as aggregated statistics. Compared to arithmetic means, SMGs avoid the over-representation of biased outlier points. The SGM of values $v_1, ..., v_N \geq 0$ with shift $s \geq 0$ is defined as

$$\left(\prod_{i=1}^{N}(v_i + s)\right)^{1/N} - s.$$

Given an SMBP problem instance, let $\underline{v}$ be a dual lower bound and $\overline{v}$ be a primal upper bound found by a solver. The relative dual gap in percentage is defined as:

$$\delta_d := \frac{\overline{v} - \underline{v}}{\overline{v}} \times 100.$$

A smaller relative dual gap indicates better performance.

Let v^a be the value of the solution found by the greedy min-utilization algorithm, which is communicated to all solvers as a warm start. The closed primal bound is defined as:

$$\delta_p := \frac{v^a - \overline{v}}{\max(\overline{v} - \underline{v}^*, 1e^{-6})} \times 100,$$

where $\underline{v}^*$ is the largest dual bound found among all solvers. A larger closed primal gap means better performance.

We report the following performance metrics for each instance tested by each solver and compute the SGMs of the benchmarks:

1. t: the total running time in CPU seconds, with a shifted value set to 1;
2. $\delta_d\%$: the relative dual gap in percentage, with a shifted value set to 1%;
3. $\delta_p\%$: the closed primal bound in percentage, with a shifted value set to 1%;
4. #N: the number of nodes of the search tree, with a shifted value set to 1;
5. #C: the number of columns generated, with a shifted value set to 1;
6. E%: the percentage of columns generated by the exact pricing algorithm, with a shifted value set to 1%;
7. $\tau\%$: the relative dual gap in the percentage of a pricing problem solved by an exact algorithm, with a shifted value set to 1%;
8. $t_p\%$: the ratio between pricing time and total solving time in percentage, with a shifted value set to 1%.

Metrics (1)-(4) refer to master problems and are available to all solvers. Metrics (5)-(8) refer to pricing problems and are not available for the BSOCP-BC, while metric (6) is 100% for the DW-PWL and DW-BC.

Benchmarks	Solvers	Problem statistics						Pricing statistics			
		t	$\delta_d\%$	$\delta_p\%$	#N	#S	#I	#C	E%	$\tau\%$	$t_p\%$
CloudSmall ($\|\mathcal{N}\| = 100$)	BSOCP-BC	1452	15.8	0.0	26601	18	0	-	-	-	-
	DW-BC	2129	11.4	0.9	21	20	17	1373	100	3.56	99
	DW-PWL	633	2.4	2.7	66	61	32	1869	100	0.01	99
	DW-Hybrid	330	2.0	3.4	127	65	36	3485	18	0.01	96
CloudMedium ($\|\mathcal{N}\| = 400$)	BSOCP-BC	3600	100.0	0.0	0	0	0	-	-	-	-
	DW-BC	3600	39.0	0.1	2	0	4	861	100	0.39	98
	DW-PWL	3600	17.2	0.4	1	0	10	3372	100	0.01	91
	DW-Hybrid	3600	11.8	0.6	12	0	15	6879	9	0.04	73
CloudLarge ($\|\mathcal{N}\| = 1000$)	BSOCP-BC	3600	100.0	0.0	0	0	0	-	-	-	-
	DW-BC	3600	59.6	0.0	2	0	0	741	100	0.04	89
	DW-PWL	3600	43.1	0.2	1	0	6	2105	100	0.01	63
	DW-Hybrid	3600	34.2	0.4	1	0	11	4257	4	0.01	8

Table 7.2 Aggregated statistics of the main computational results

7.5.3 Comparative analysis of results

The main computational results are summarized in Table 7.2. For each benchmark, we report the SGM statistics of the performance metrics, the number of instances solved (denoted by #S), and the number of instances with improved primal bounds (denoted by #I). We also report a computational test of adaptive selection of break points in 7.5.3. Next, we analyze the main computational results by comparing the solvers.

We first compare the compact BSOCP formulation of SMBP (7.4) with the set cover formulation (7.5). So, we evaluate the performance of BSOCP-BC and DW-BC. For all the benchmarks, DW-BC achieves smaller dual gaps than BSOCP-BC. For small instances, DW-BC also explores a smaller number of nodes of the search tree. These observations agree with Prop. 7.2 that the continuous relaxation of the set covering formulation is stronger than the continuous relaxation of the compact formulation. The number of nonlinear integer constraints in the compact formulation increases with the number of bins. For medium instances, BSOCP-BC cannot even finish the root node computation of the compact formulation. However, DW-BC can prove a dual gap or improve primal solutions by solving the set cover formulation. Although the two formulations are insufficient to tackle medium or large instances, the compact formulation is better overall than the set cover formulation. Then, we will solely examine algorithms that tackle the set cover formulation in the following.

We next evaluate our core innovation to solve the pricing subproblems: the PWL relaxation and its associated combined formulation (7.18). So, we compare DW-BC with DW-PWL. DW-BC just calls `CPLEX` to solve the BSCOP formulation (7.10) of pricing subproblems, while DW-PWL uses a tailored branch-and-cut algorithm to solve the combined formulation (7.18). Looking at the problem statistics for all the benchmarks, we find that DW-PWL significantly reduces the master problem' dual gap than DW-BC. Especially for small instances, DW-PWL achieves nearly five times improvement to DW-BC. More details can be found in pricing statistics. DW-PWL can solve pricing subproblems to optimality (pricing gap on average is 0.01%) in a short time and thus produce much more columns than DW-BC. Especially for large instances, we find that combined formulation (7.18) is still solvable. The overall quality of the combined formulation for submodular knapsack outperforms that of the BSOCP formulation (7.10).

We examine the hybrid pricing strategy, which replaces the computationally expensive exact pricing with computationally cheap heuristic pricing when the exact pricing is not in

need. So, we compare DW-PWL with DW-Hybrid. Looking at the problem statistics, the hybrid pricing strategy achieves smaller dual gaps, especially for large instances; it also saves computational time for small instances. Looking at the pricing statistics, the hybrid pricing strategy can generate twice the number of columns than the exact pricing, for all the instances. As a byproduct, with more columns, `SCIP` can find more improved primal solutions. We find that the hybrid pricing strategy gives rise to consistent improvement.

We look at the column selection heuristic. So, we compare DW-Hybrid with DW-Hybrid*. The column selection heuristic can find more improved solutions.

Finally, we summarize our computational results. The set cover formulation is better than the compact formulation regarding scalability, although both formulations are unsolvable for medium and large instances. Our techniques can improve the column generation procedure for the set cover formulation. Regarding pricing subproblems, a dense BSOCP constraint might be reformulated as a submodular knapsack constraint, so the good performance of PWL relaxations suggests that PWL relaxations can provide strong MILP relaxations for dense BSOCP constraints. This finding can also help solve other BSOCP problems. The hybrid pricing strategy uses a hint from the Farley bound, so it reduces computational time and is applicable for other column generation problems. As for benchmarks, `CloudSmall` is a suitable testbed for comparing solvers, `CloudMedium` is suitable for testing the pricing algorithms, and `CloudLarge` is still too big to handle.

Non-equidistant breakpoints

According to Thm. 7.8 in Sect. 7.4.4, the optimal breakpoints under the ℓ_∞ error form an equidistant partition of $[\underline{w}, \overline{w}]$. In this section, we investigate whether adaptive non-equidistant breakpoints can improve the DW-PWL. There are many possibilities for non-equidistant breakpoints, and we propose a regression approach using the previous pricing information.

We recall that the DW-PWL solver adds a lazy cut at each infeasible solution $\hat{x}$. Let $\hat{w} := \sum_{i \in \mathcal{N}'} a'_i \hat{x}_i$ be the corresponding value of variable w, and we call it an infeasible w-value. For an objective coefficient vector c', let $[w_\ell(c'), w_u(c')]$ be the range of the set of infeasible w-values, which are recorded during the PWL-B&C algorithm for every pricing problem. Our intuition is that for a new pricing problem with an objective coefficient vector c, one may reduce the search space by concentrating breakpoints to the range $[w_\ell(c), w_u(c)]$, because this refines the PWL relaxation in that region. Usually $[w_\ell(c), w_u(c)]$ is unknown, so one can only use a predication range $[w'_\ell(c), w'_u(c)]$. Given the fixed number of breakpoints of $\mathcal{B}$, with this limited resource, we use the $k$nn regression approach to learn $[w'_\ell(c), w'_u(c)]$ and concentrate a subset of of $\mathcal{B}$ to $[w'_\ell(c), w'_u(c)]$.

The $k$nn regression is as follows. Let T be the number of pricing iterations, and we use a list $\{[w_\ell(c^t), w_u(c^t)]\}_{1 \le t \le T}$ to record the set of intervals, where $w_\ell(c^t), w_u(c^t)$ are the lower and upper bounds of the set of infeasible w-values in the t-th pricing problem. For the new objective coefficient vector c, we sort the list in an increasing order w.r.t. the ℓ_2-norm distances between $\{c^t\}_{1 \le t \le T}$ to c. The predicted range $[w'_\ell(c), w'_u(c)]$ is as follows:

$$w'_\ell(c) = \sum_{1 \le t \le k} w_\ell(c^t)/k, w'_u(c) = \sum_{1 \le t \le k} w_u(c^t)/k.$$

Recall that there are in total h breakpoints in the range $[\underline{w}, \overline{w}]$. Let $r := (w'_u(c) - w'_\ell(c))/(\overline{w} - \underline{w})$ be the range ratio. Then, given a concentration scale $s > 1$, we put $h * r * s$ number of

	$k=1$				$k=3$			$k=5$		
	$s=1$	$s=1.5$	$s=1.5$	$s=2.5$	$s=1.5$	$s=2$	$s=2.5$	$s=1.5$	$s=2$	$s=2.5$
$\delta_d\%$	35.29	35.58	35.66	35.36	35.28	36.28	35.29	35.39	34.95	35.29
$\#C$	2134.12	2120.03	2097.46	2123.38	2118.57	2091.97	2148.48	2111.01	2155.95	2134.12

Table 7.3 Master and pricing problem statistics of different configurations

breakpoints equidistantly in $[w'_\ell(c), w'_u(c)]$, and $h*(1-r*s)$ number of breakpoints equidistantly in the remaining breakpoint region. This concentration results in a PWL relaxation with a better approximation in $[w'_\ell(c), w'_u(c)]$. Therefore, we hope that the adaptive PWL relaxation could use the previous pricing information.

To understand the performance of the $k$nn regression approach, we have several configurations with combinations of $k \in \{1, 3, 5\}$ and $s \in \{1.5, 2, 2.5\}$. We note that with $k = 1, s = 1$, the configuration is exactly the DW-PWL solver. To test these configurations, we generate two new benchmarks with $|\mathcal{N}| \in \{500, 900\}$, each containing 36 instances. The aggregated computational results are presented in Table 7.3, which displays SGMs of the relative dual gap of master problems, and the number of generated columns.

The non-equidistant breakpoints do not lead to an improvement of the algorithm. In most cases, $k$nn regression approach is even worse than the equidistant breakpoint approach. Therefore, finding good breakpoints is a complex task.

7.6 Conclusion

We develop a PWL-B&C algorithm for solving pricing submodular knapsack problems. The PWL-B&C algorithm is more efficient than the conventional LP-B&C algorithm implemented in `CPLEX` for the pricing submodular knapsack problems. The PWL-B&C algorithm can also be extended to solve the multiple submodular knapsack problems. For general MINLP problems, if a nonlinear constraint can be reformulated into a linear part and a univariate concave part, then the univariate concave part can be convexified by the PWL relaxation.

Our hybrid pricing strategy applies to the column generation procedure, where the master problems are in set cover formulations, as long as there are fast pricing heuristics. This pricing strategy is helpful for large instances. As a future study, we can apply this strategy to solve the DW decomposition of the capacitated vehicle routing problem, for which the pricing problem is complex.

The primary efforts of this chapter are solving pricing submodular knapsack subproblems with conflicts via PWL relaxations and speeding up column generation via a hybrid pricing strategy. There is still much room for improvement in future studies. Since the submodular knapsack with conflicts is solved multiple times with different parameters, the information of previous column generation iterations can be leveraged statistically to reduce the search space of pricing subproblems.

On the other hand, commonly known techniques for branch-and-price algorithms are generally helpful. Combining our techniques with other advanced elements from general-purpose framework [250] could be also useful. For example, we can use stabilization techniques to speed up the convergence of the column generation or use cutting planes to tighten the relaxation of the master problem.

Chapter 8

Branch-and-price for coding-aware routing in wireless networks

8.1 Introduction

Multi-hop wireless sensor networks (WSNs) support many applications requiring wireless communication on various platforms. We can mention unmanned aerials vehicles (UAVs) [231, 244], flying taxis (Vertical Take-Off and Landing) [248], Internet-of-Things (IoT) sensor devices for monitoring [114, 307], connected healthcare [161, 308], agricultural monitoring and various emerging smart city and smart mobility deployments [213, 315], whose components are connected via WSNs. In all these cases, energy efficiency and traffic optimization are key challenges. In the upcoming IoT landscape [10, 258], smart devices are expected to be widely deployed everywhere over the world [146]. Current statistics indicate that billions of IoT devices are already deployed and connected in 2020, and they are expected to grow substantially in the future [13]. WSNs are having significant and growing effects on energy consumption and environmental issues. The research community faces solving optimization problems that will shape the connectivity of billions of devices with significant energy issues. Most WSN technologies and deployments rely on single-hop communications. Multi-hop communication in WSNs would increase network capacity and coverage without requiring new infrastructure. Extensions to multi-hop communication especially in IoT-related technologies are attracting research and industrial interest [158, 273].

In this chapter, we study the problem of energy efficiency in multi-hop WSNs, namely wireless unsplittable multi-commodity flow with network coding (wUMCFC). We use the DW relaxation for the resulting MILP problem and propose an exact branch-and-price algorithm.

The lifetime of the network strongly depends on the energy level of its devices. Each node in the WSN has limited wireless communication capabilities and energy source (usually a battery). Due to the small sizes of the sensors, the batteries are also small and the available energy is limited. The optimal management of energy is necessary to ensure a long network lifetime. A classification of the main used batteries in the WSN is given in [285].

Routing strategies have a major impact on the total energy consumption of networks. In multi-hop WSNs each communication is routed through a single path, i.e., unsplittably from its source node to its target node [25–27]. Hence the intermediate nodes in the path are not in charge of computing, for each data packet, the next hop-node.

Unsplittable routing allows intermediate nodes to use less memory, speeds up packet handling processes, minimizes loss rates and enables better quality of service (QoS). However unsplittable routing induces very complex combinatorial optimization problems [37, 44, 190, 298]. Splittable routing is very complex to apply in a wireless context. This would require sophisticated protocols and more intelligence in the network components. For more details on routing protocols in WSNs, we refer to [106].

Network coding allows intermediate nodes of the network to encode several packets into a single packet, and then broadcast, i.e., transmit simultaneously to all neighbours only once [238]. Broadcasting is the term used to describe communication where the data packets are sent from one node to all other connected nodes; a single sender transmits data, and the data is sent to all connected receivers. Network coding reduces energy consumption in WSNs by reducing the number of transmissions required in a network to carry traffic between the set of sources and the set of destinations [4, 52, 88, 182, 200]. The deployment of network coding and broadcasting realizes significant benefits in terms of resource and energy management and improves a network's throughput, efficiency and scalability, as well as resilience to attacks and eavesdropping [140].

Interference has a significant impact on energy consumption, networking operation and performance. Wireless communication relies on shared communication media, that can be accessed by several devices, close to each other, at the same time. Interference is caused by simultaneous transmissions between these devices. Several strategies and models are developed in the literature to handle interference in WSNs [309].

8.1.1 Literature review

In [240], the authors proposed a linear programming model to compute optimal routing, that minimizes the number of data transmissions, without taking into account the interference. In [198], a MILP was proposed to optimize the routing with network coding, but the effect of energy saved by network coding was not considered.

This article presents mathematical formulations of the wUMCFC problem that integrate interference and network coding. A column generation approach and a branch-and-price framework are then described. The proposed models are based on the unsplittable multi-commodity flow (UMCF) problem formulations. UMCF is one of the well-known $\mathcal{NP}$-hard problems in combinatorial optimization [24, 126, 190, 191]. The problem addressed in this chapter is more complex since it generalizes the UMCF problem with additional coding and interference constraints. Extensive research is required to adapt mathematical programming approaches and develop efficient algorithms to solve it. Multi-commodity flow models are developed for several network optimization problems since they lead to modeling complex technical constraints [14, 43, 50, 151, 199, 298]. The technical constraints can be the unsplittable routing or resource sharing constraints [24, 191, 168]. Another advantage of the multi-commodity flow formulations is that they can be solved efficiently by decomposition and column generation methods [37, 152, 168, 215, 251, 297].

8.1.2 Contribution

The first contribution of this chapter is our quantitative analysis and modelling of interference, network coding and energy consumption in the context of WSNs. We propose the new problem

wUMCFC and new models to integrate interference and network coding. Given the network topology, source-target traffic demands, transmissions capacities, and channels' interference, the wUMCFC problem seeks to minimize the energy cost of data transmission in multi-hop WSNs and find an optimal unsplittable routing. The wUMCFC problem incorporates network coding to reduce data transmission, save energy consumption and improve quality of service (QoS). Interference is modelled using adapted capacity constraints, which are defined over the clique set of an undirected conflict graph. We show that the wUMCFC problem is an $\mathcal{NP}-$hard problem.

The second contribution involves our formulations of the wUMCFC problem. The first class of models are compact edge-based formulations, which consist of a mixed-Boolean quadratic programming (MBQP) formulation and two MILP formulations. The two MILP models are respectively the edge balance formulation and the edge linearization formulation. We study the strength of these two MILP formulations. The second class of models is a DW reformulation of the edge balance model, namely a path-based formulation.

To solve the path-based formulation efficiently, we develop a column generation approach and a branch-and-price (B&P) algorithm [15, 37, 152]. The algorithm is implemented in a new open source solver *wUMCFC*. This yields our third contribution. In our B&P algorithm, the pricing problem is reduced to a shortest path problem in an extended graph. Although the edge weights of the extended graph can be negative, we prove that the cycles of the extended graph have positive costs. Therefore, a shortest path in the extended graph can be calculated in polynomial time. We show that, under our reduction, the path generated in the original graph is always a simple path. We perform a computational study on realistic problem instances with an analysis of the performance of the B&P algorithm and the effect of the network coding.

8.1.3 Outline the chapter

This article is organized as follows: In Section 8.2, we introduce the classical UMCF problem formulation and notation. We present the three important aspects of the wUMCFC problem: energy consumption, clique capacity constraints and network coding. We then analyze the complexity of the wUMCFC problem. In Section 8.3, we present and compare the compact edge-based formulations and the path-based formulation. In Section 8.4, we propose a new algorithm to solve the LP relaxation of the path-based formulation. We discuss the column generation approach, the pricing problem and the B&P algorithm. In Section 8.5, we describe branching rules to enforce the integrality of path variables. In Section 8.6, we perform two experiments. The first experiment shows that the B&P algorithm for the path-based formulation outperforms the MILP solver CPLEX for the edge balance formulation. The second experiment demonstrates that the network coding mechanism can decrease the energy cost significantly. Conclusions are drawn in Section 8.7 along with the prospect of future research.

8.2 Models and notation

The network topology is represented by a bi-directed graph $G = (V, E)$, where V denotes the set of nodes corresponding to wireless transmission devices and E denotes the set of transmission links that can be used to route the traffic.

Unsplittable traffic demands are denoted by a set D of source-target node pairs (s, t).

We define the following notation of data and parameters:

C_{ij}: the transmission capacity of the edge (i, j). It measures the number of data packets that can be sent through the channel (i, j) per unit of time.

β_{ij}: the energy cost parameter for flow transmission along the edge (i, j). It measures the energy cost to transmit a unit data packet per unit of time.

d^{st}: the traffic demand from the source node s to the target node t for $st \in D$.

Decision variables:

x_{ij}^{st} : binary variable indicating whether demand st is routed on the edge (i, j), for $st \in D$ and $(i, j) \in E$.

The occupancy time ratio **(OTR)** is an important concept in wireless communication. The OTR measures the ratio that the channel along (i, j) is transmitting data per unit of time. Hence, the forwarding node i consumes energies during the transmission time. For each edge $(i, j) \in E$, its OTR is defined as the total flow per unit of time divided by its capacity, i.e.,

$$\frac{\sum_{st \in D} d^{st} x_{ij}^{st}}{C_{ij}}. \tag{8.1}$$

The UMCF problem aims to find a unique routing path for each demand that minimizes the total energy cost under capacity and demand constraints.

Before introducing network interference and network coding, let us recall the ILP formulation of the classical minimum cost flow (UMCF) problem:

$$\begin{aligned}
\min \quad & z = \sum_{st \in D} \sum_{(i,j) \in E} \beta_{ij} d^{st} x_{ij}^{st} && && (8.2.0)\\
& \sum_{j:(i,j) \in E} x_{ij}^{st} - \sum_{j:(j,i) \in E} x_{ji}^{st} = 0, && \forall i \in V - \{s, t\},\ \forall st \in D, && (8.2.1)\\
& \sum_{(s,i) \in E} x_{si}^{st} - \sum_{(i,s) \in E} x_{is}^{st} = 1, && \forall st \in D, && (8.2.2)\\
& \sum_{(i,t) \in E} x_{it}^{st} - \sum_{(t,i) \in E} x_{ti}^{st} = 1, && \forall st \in D, && (8.2.3)\\
& \sum_{st \in D} \frac{d^{st}}{C_{ij}} x_{ij}^{st} \leq 1, && \forall (i, j) \in E, && (8.2.4)\\
& x_{ij}^{st} \in \{0, 1\}, && \forall (i, j) \in E,\ \forall st \in D.
\end{aligned} \tag{8.2}$$

Objective function z (8.2.0): energy consumption of data transmission per unit of time.

Flow conservation constraints (8.2.1) to (8.2.3): flow conservation constraints at each node.

Edge capacity constraint (8.2.4): the total flow on an edge should not exceed the available transmission capacity. This constraint stipulates that the OTR of an edge must be at most 1.

The mathematical formulation of the wUMCFC problem requires the addition of new constraints and new variables to the classical UMFC. This new problem integrates interference and coding mechanisms.

In the following subsections, we present three important factors of the wUMCFC problem: energy consumption, clique capacity constraints, and network coding. We also analyze the complexity of the wUMCFC problem.

8.2.1 Energy consumption

This chapter considers the energy consumption induced by data transmissions via active devices in the network.

A node $i \in V$ is active when it is transmitting data. We introduce an energy cost parameter β_i that measures the cost of energy consumed by an active node i per unit of time. This parameter depends on the characteristics of the communication device corresponding to the node i.

For $(i,j) \in E$, let f_{ij} be the flow on the edge, i.e., the number of data packets to transmit from node i to node j per unit of time. Recall that the OTR $\frac{f_{ij}}{C_{ij}}$ is the time ratio that node i is transmitting data along the channel (i,j). The energy cost per unit of time on the channel (i,j) is $\frac{f_{ij}}{C_{ij}}\beta_i$. During the remaining part of a time unit $1 - \frac{f_{ij}}{C_{ij}}$, energy is not consumed because the node i does not transmit data through (i,j).

The energy cost parameter β_{ij} of (i,j), follows as:

$$\beta_{ij} = \frac{\beta_i}{C_{ij}} \tag{8.3}$$

8.2.2 Clique capacity constraint

The difference between wireless and wired networks lies mainly in the use of communication channels and transmission technologies [314]. In wired networks, the capacity of one channel is not affected by the data transmissions of any other channels. In WSNs, channels inherently share the same communication space, and the interference between channels in a neighborhood would decrease their capacities. When a channel transmits a data packet, it consumes the capacity of its neighbours. Interference reduces transmission capacities significantly [28].

We model interference using capacity constraints, which are defined over the clique set of an undirected conflict graph $G_c = (N, L)$ where $N = E$.

The nodes of the conflict graph G_c are the edges of the network graph G. The links of G_c represent interference between the edges of G which cannot transmit data at the same time.

If two edges in G are in interference, then a link between their corresponding nodes is added to G_c. The graph G_c is constructed incrementally with an initial empty link set L, as follows:

1. $N = E$ and $L = \emptyset$.
2. If $(i,j) \in N$ and $(k,l) \in N$ are under interference then add the link $\{(i,j),(k,l)\}$ to L: $L = L \cup \{(i,j),(k,l)\}$.

Interference can be modeled in various ways [164]. We introduce the $n-$dist interference model to construct the conflict graphs in our experiments. The $n-$dist model extends single-node and two-node models proposed in [83].

Let $p = (v_1, \ldots, v_h)$ be a simple path in the graph G, where v_t ($t \in \{1, \ldots, h\}$) is the node in the path p. We define the length of p as the number of nodes h in the path. For $i,k \in V$, we define $\mathrm{dist}(i,k)$ as the length of a shortest path from i to k or from k to i, i.e., a path with the minimum number of nodes. For (i,j) and $(k,l) \in E$, the distance $\mathrm{dist}\,((i,j),(k,l))$ is defined as the length of a shortest path between any pair of $\{i,k\},\{i,l\},\{j,k\}$, and $\{j,l\}$, i.e., $\mathrm{dist}\,((i,j),(k,l)) = \min_{t_1 \in \{i,j\}, t_2 \in \{k,l\}}\{\mathrm{dist}(t_1,t_2), \mathrm{dist}(t_2,t_1)\}$. Hence, $\mathrm{dist}\,((i,j),(k,l))$ measures the minimum distance between the tail and end nodes of edges (i,j) and (k,l) .

Under the $n-$dist model, (i,j) and (k,l) are in interference if $\mathrm{dist}\,((i,j),(k,l)) \leq n$.

For example, Figure (8.1a) shows a network of five nodes and Figure (8.1b) shows its conflict graph constructed via the $2-$dist interference model. The $2-$dist interference model corresponds to the single-node model in [83]. This model considers that a node interferes with its neighbors.

The distance between (1,2) and (4,5) is 3, so they are not connected via any link in G_c.

The distance between (1,2) and (3,4) is 2, so they are connected via one link in G_c. If two edges in G are in interference, then a link between their corresponding nodes is in G_c. Hence we can check whether two edges share capacity by the adjacency of their corresponding nodes in G_c.

(a) Network with 5 nodes

(b) Conflict graph

Figure 8.1 Interference model

We develop a capacity-sharing model in a wireless communication context from [165].

For edge $(i,j) \in E$, let f_{ij} be the flow on this edge; recall that $\frac{f_{ij}}{C_{ij}}$ is the OTR of this edge. Two edges connected in the conflict graph cannot transmit at the same time otherwise these transmissions will fail.

Let m be a subset of edges of G such that the corresponding nodes in G_c are a clique. Then every edge of m shares OTR with other edges in m.

The sum of OTRs of edges in m should be at most 1 and the clique capacity constraint follows as:

$$\sum_{(i,j)\in m} \frac{f_{ij}}{C_{ij}} \leq 1. \tag{8.4}$$

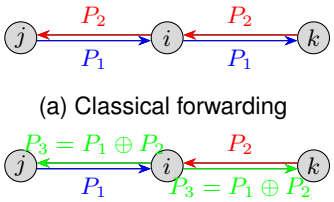
For any two clique sets m_1 and m_2, such that $m_1 \subset m_2$, it follows that the clique capacity constraint over m_1 is dominated by the clique capacity constraint over m_2:

$$\sum_{(i,j)\in m_1} \frac{f_{ij}}{C_{ij}} \leq \sum_{(i,j)\in m_2} \frac{f_{ij}}{C_{ij}}. \tag{8.5}$$

Therefore, non-dominated constraints are defined over maximal cliques. The dominance relation over clique capacity constraints is equivalent to set inclusion over the corresponding cliques. Then, it suffices to consider only non-dominated constraints induced by maximal cliques in the model.

We denote by M the set of maximal cliques of G_c.

In Figure (8.1b), $m_1 = \{(1,2),(2,3)\}$ is a clique set, but $m_2 = \{(1,2),(2,3),(3,4)\}$ is the maximal clique set including it, therefore the capacity constraint is expressed only on the maximal clique m_2.



(a) Classical forwarding

(b) Forwarding with network coding

Figure 8.2 Network coding

There are two maximal cliques of G_c in Figure (8.1b), and hence

$$M = \{\{(1,2),(2,3),(3,4)\},\{(2,3),(3,4),(4,5),(5,4)\}\}.$$

The model contains two capacity constraints:

$$\sum_{(i,j)\in\{(1,2),(2,3),(3,4)\}} \frac{f_{ij}}{C_{ij}} \leq 1,$$
$$\sum_{(i,j)\in\{(2,3),(3,4),(4,5),(5,4)\}} \frac{f_{ij}}{C_{ij}} \leq 1.$$

The following section is dedicated to presenting the network coding technique in WSN.

8.2.3 Network coding

Network coding allows intermediate nodes to combine data packets into a single packet before broadcasting. It is a networking technique where operations, which in practice tend to be algebraic algorithms, are performed on data to reduce the number of transmissions and energy consumption. Broadcasting is the term used to describe communication where the data packets are sent from one node to all other connected nodes.

Figure (8.2) illustrates the network coding mechanism by a triple of nodes; node $j \in V$ and node $k \in V$ send packets P_1 and P_2 to each other through the intermediate node $i \in V$.

The classical forwarding scheme in Figure (8.2a) uses four transmissions to send P_1 and P_2. Node j (resp. k) sends its packet P_1 (resp. P_2) to node i, and node i sends P_1 to k and P_2 to j, separately.

Figure (8.2b) shows that with network coding, only three transmissions are needed, and hence the energy cost at the device i is saved. First node j and node k record and send their packets to node i. Then node i encodes these two data packets by XOR operation to obtain an encoded packet $P_3 := P_1 \oplus P_2$, where $\oplus$ is the bit-wise Boolean addition. Node i broadcasts the encoded packet P_3 to node j and node k simultaneously. Finally, node j and node k decode data of P_3 by XOR-ing with recorded packets, $P_1 = P_3 \oplus P_2$ and $P_2 = P_3 \oplus P_1$.

If $(j,i),(i,k),(k,i)$, and (i,j) are in E, then there is a coding opportunity on the node i called the three-node pattern.

The opportunity set is defined as follows:

$$\Lambda^i_{\{k,j\}} \coloneqq \{(j,i),(i,k),(k,i),(i,j)\}\,. \tag{8.6}$$

For $\Lambda^i_{\{k,j\}} \subset E$, let f_{jik} be the flow along $j \to i \to k$, and let f_{kij} be the flow along $k \to i \to j$.

If f_{jik} and f_{kij} are non-zero, then network coding can be applied, and the node i could encode the opposite flows f_{jik} and f_{kij}.

We define $u^i_{\{k,j\}}$ as the flow encoded by node i, and it measures the number of data packets in f_{jik} and f_{kij} that node i could code per unit of time.

Since the maximum encoded data cannot exceed the number of data arriving at a node i, $u^i_{\{k,j\}}$ must satisfy the following inequality:

$$u^i_{\{k,j\}} \leq \min(f_{jik}, f_{kij}). \tag{8.7}$$

The next sections explain how the network coding increases the capacity of the network and decreases the energy cost.

Effects on clique capacity constraints

Network coding decreases the left-hand side of the capacity constraints since it reduces the occupancy time rate (OTR). Without network coding, the OTRs sum for transmitting f_{jik} and f_{kij} is:

$$\frac{f_{jik}}{C_{ji}} + \frac{f_{jik}}{C_{ik}} + \frac{f_{kij}}{C_{ki}} + \frac{f_{kij}}{C_{ij}}. \tag{8.8}$$

Let $T^i_{\{k,j\}}u^i_{\{k,j\}}$ be the OTR of the broadcasted flow $u^i_{\{k,j\}}$ by network coding, where $T^i_{\{k,j\}}$ is a network transmission parameter. The broadcasting time must be at least the time of one of the two separate transmissions:

$$T^i_{\{k,j\}} \geq \max\{\frac{1}{C_{ik}}, \frac{1}{C_{ij}}\}, \tag{8.9}$$

The remaining parts of $f_{jik} - u^i_{\{k,j\}}$ and $f_{kij} - u^i_{\{k,j\}}$ are transmitted by the classical forwarding scheme.

The OTR with network coding follows as:

$$\begin{aligned} &\frac{f_{jik}}{C_{ji}} + \frac{f_{jik} - u^i_{\{k,j\}}}{C_{ik}} + \frac{f_{kij}}{C_{ki}} + \frac{f_{kij} - u^i_{\{k,j\}}}{C_{ij}} + T^i_{\{k,j\}}u^i_{\{k,j\}} \\ =&\frac{f_{jik}}{C_{ji}} + \frac{f_{jik}}{C_{ik}} + \frac{f_{kij}}{C_{ki}} + \frac{f_{kij}}{C_{ij}} - (\frac{1}{C_{ij}} + \frac{1}{C_{ik}} - T^i_{\{k,j\}})u^i_{\{k,j\}}. \end{aligned} \tag{8.10}$$

Let $C^i_{\{k,j\}}$ be the increased capacity over the three-node pattern $\Lambda^i_{\{k,j\}}$ with network coding. $C^i_{\{k,j\}}$ measures the number of data packets that will no longer be transmitted on $\Lambda^i_{\{k,j\}}$ due to network coding:

$$\frac{1}{C^i_{\{k,j\}}} \coloneqq \frac{1}{C_{ij}} + \frac{1}{C_{ik}} - T^i_{\{k,j\}}. \tag{8.11}$$

From the bound on $T^i_{\{k,j\}}$, the following bound on $\frac{1}{C^i_{\{k,j\}}}$ is obtained:

$$\frac{1}{C^i_{\{k,j\}}} \leq \min\left\{\frac{1}{C_{ij}}, \frac{1}{C_{ik}}\right\}. \tag{8.12}$$

Equivalently, $C^i_{\{k,j\}} \geq \max\{C_{ij}, C_{ik}\}$.

As a result, the OTR can be rewritten as:

$$\frac{f_{jik}}{C_{ji}} + \frac{f_{jik}}{C_{ik}} + \frac{f_{kij}}{C_{ki}} + \frac{f_{kij}}{C_{ij}} - \frac{u^i_{\{k,j\}}}{C^i_{\{k,j\}}}. \tag{8.13}$$

The OTR decrease within $\Lambda^i_{\{k,j\}}$ is $\frac{u^i_{\{k,j\}}}{C^i_{\{k,j\}}}$.

Let $m \in M$ be a maximal clique of G_c. The clique capacity constraint is obtained by summing up the OTRs decreased by network coding, for all opportunity sets in m:

$$\sum_{(i,j)\in m} \frac{f_{ij}}{C_{ij}} - \sum_{\Lambda^i_{\{k,j\}}\subset m} \frac{u^i_{\{k,j\}}}{C^i_{\{k,j\}}} \leq 1. \tag{8.14}$$

Inequalities (8.14) and (8.7) define the clique capacity constraint with network coding.

In our experiments, we set $T^i_{\{k,j\}}$ to its lower bound $\max\{\frac{1}{C_{ki}}, \frac{1}{C_{ij}}\}$, and correspondingly set $C^i_{\{k,j\}}$ to $1/(\frac{1}{C_{ij}} + \frac{1}{C_{ik}} - T^i_{\{k,j\}}) = \min\{C_{ki}, C_{ij}\}$.

In practice, the OTR of broadcasting $u^i_{\{k,j\}}$ can be at most equal to the lower bound $\max\left\{\frac{u^i_{\{k,j\}}}{C_{ki}}, \frac{u^i_{\{k,j\}}}{C_{ij}}\right\}$; a higher $C^i_{\{k,j\}}$ can be set accordingly.

Effects on energy consumption

Network coding allows us to reduce the energy cost and decrease the objective function value.

Without network coding, the energy consumption to send the flow $u^i_{\{k,j\}}$ from node i to node j and to node k separately is $u^i_{\{k,j\}}(\beta_{ij}+\beta_{ik})$. Let $\beta^i_{\{k,j\}}u^i_{\{k,j\}}$ be the cost of broadcasting $u^i_{\{k,j\}}$ by network coding, where the energy cost parameter $\beta^i_{\{k,j\}}$ measures the energy consumption to send a unit packet of $u^i_{\{k,j\}}$ per unit time by broadcasting. The transmission cost $\beta^i_{\{k,j\}}u^i_{\{k,j\}}$ is equal to the OTR of broadcasting $u^i_{\{k,j\}}$ times β_i, i.e., $\beta^i_{\{k,j\}}u^i_{\{k,j\}} = u^i_{\{k,j\}}T^i_{\{k,j\}}\beta_i$. Dividing by $u^i_{\{k,j\}}$, it follows that $\beta^i_{\{k,j\}} = \beta_i T^i_{\{k,j\}}$.

Therefore, the energy cost of sending flow f_{jik} and f_{kij} is reduced to

$$\begin{aligned}&\beta_{ji}f_{jik} + \beta_{ik}(f_{jik} - u^i_{\{k,j\}}) + \beta_{ki}f_{kij} + \beta_{ij}(f_{kij} - u^i_{\{k,j\}}) + \beta^i_{\{k,j\}}u^i_{\{k,j\}}\\ =&\beta_{ji}f_{jik} + \beta_{ik}f_{jik} + \beta_{ki}f_{kij} + \beta_{ij}f_{kij} - u^i_{\{k,j\}}(\beta_{ik} + \beta_{ij} - \beta^i_{\{k,j\}}).\end{aligned} \tag{8.15}$$

Denote $\tau^i_{\{k,j\}} = \beta_{ij}+\beta_{ik}-\beta^i_{\{k,j\}}$, so the energy cost saved by network coding is $\tau^i_{\{k,j\}}u^i_{\{k,j\}}$. The energy saving and the increased capacity are coupled according to Section 8.2.1:

$$\tau^i_{\{k,j\}} = \beta_i\left(\frac{1}{C_{ij}} + \frac{1}{C_{ik}} - T^i_{\{k,j\}}\right) = \beta_i\frac{1}{C^i_{\{k,j\}}}. \tag{8.16}$$

It follows from Section 8.2.1 and the inequality (8.12) that:

$$0 \leq \tau^i_{\{k,j\}} \leq \beta_i \min\left\{\frac{1}{C_{ij}}, \frac{1}{C_{ik}}\right\} \leq \min\left\{\beta_{ij}, \beta_{ik}\right\}. \tag{8.17}$$

8.2.4 Complexity analysis

The wUMCFC problem is formulated as an UMCF problem with additional network coding and clique capacity constraints. The wUMCFC problem can be reduced to a single-source unsplittable flow problem [126, 190, 191, 222] by considering a single source demand, where clique sets are singletons, and the coding variables are fixed to zero.

The single-source unsplittable flow problem is $\mathcal{NP}-$hard, hence the wUMCFC problem is also $\mathcal{NP}-$hard.

8.3 Mathematical Formulations

In this section, we present mathematical programming formulations of the wUMCFC problem: the compact edge-based formulations and the path-based formulation. We first define the problem notation.

We denote by wUMCF the model derived from the UMCF problem by including network coding variables and constraints. The edge capacity constraints are reformulated as the clique capacity constraints defined in Section 8.2.3. Moreover, the objective function z_- is obtained by subtracting from z the energy cost saved by network coding z_c; where $z_c := \sum_{\Lambda^i_{\{k,j\}} \subset E} \tau^i_{\{k,j\}} u^i_{\{k,j\}}$.

An abbreviation for edge (resp. path) formulation, i.e., E$\#$ (resp. P), is appended at the end of the model notation, where $\#$ will be revealed subsequently. For example, wUMCFC-P denotes the path-based formulation of the wUMCFC problem.

8.3.1 Compact edge-based formulations

In this subsection, we propose the MBQP formulation, and derive two edge-based MILP formulations: the edge linearization formulation and the edge balance formulation.

MBQP formulation

The decision variables of the MBQP formulation are defined as follows:

Decision variables:

x^{st}_{ij} : binary variable indicating whether demand $st \in D$ is routed on the edge $(i,j) \in E$.

q^{st}_{jik} : real variable denoting the flow value of the demand $st \in D$ routed on the incident edges $j \to i \to k$. q^{st}_{jik} should satisfy the quadratic constraint: $q^{st}_{jik} = d^{st} x^{st}_{ji} x^{st}_{ik}$.

$u^i_{\{k,j\}}$: real coding variable denoting the value of two opposite flows along the incident edges $k \to i \to j$ and $j \to i \to k$, which would be encoded at the node i.

M is the set of the maximum cliques of the conflict graph G_c, and $\tau^i_{\{k,j\}} u^i_{\{k,j\}}$ ($\Lambda^i_{\{k,j\}} \subset E$) is the energy cost saved by network coding.

The MBQP formulation, denoted by wUMCFC-EQ, follows as:

$$
\begin{aligned}
\min \quad & z_- = \sum_{st\in D}\sum_{(i,j)\in E}\beta_{ij}d^{st}x_{ij}^{st} - \sum_{\Lambda^i_{\{k,j\}}\subset E}\tau^i_{\{k,j\}}u^i_{\{k,j\}} && && (8.18.0)\\
& \textstyle\sum_{(i,j)\in E} x_{ij}^{st} - \sum_{(j,i)\in E} x_{ji}^{st} = 0, && \forall i \in V-\{s,t\},\ \forall st\in D, && (8.18.1)\\
& \textstyle\sum_{(s,i)\in E} x_{si}^{st} - \sum_{(i,s)\in E} x_{is}^{st} = 1, && \forall st\in D, && (8.18.2)\\
& \textstyle\sum_{(i,t)\in E} x_{it}^{st} - \sum_{(t,i)\in E} x_{ti}^{st} = 1, && \forall st\in D, && (8.18.3)\\
& q_{jik}^{st} - d^{st}x_{ji}^{st}x_{ik}^{st} = 0, && \forall (j,i),(i,k)\in E,\ \forall st\in D, && (8.18.4)\\
& u^i_{\{k,j\}} - \sum_{st\in D} q_{jik}^{st} \le 0, && \forall \Lambda^i_{\{k,j\}}\subset E, && (8.18.5)\\
& u^i_{\{k,j\}} - \sum_{st\in D} q_{kij}^{st} \le 0, && \forall \Lambda^i_{\{k,j\}}\subset E, && (8.18.6)\\
& \sum_{st\in D}\sum_{(i,j)\in m}\frac{d^{st}}{C_{ij}}x_{ij}^{st} - \sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{u^i_{\{k,j\}}}{C^i_{\{k,j\}}} \le 1, && \forall m\in M, && (8.18.7)\\
& x_{ij}^{st}\in\{0,1\}, && \forall (i,j)\in E,\ \forall st\in D, &&\\
& q_{jik}^{st}\in\mathbb{R}_+, && \forall (j,i),(i,k)\in E,\ \forall st\in D, &&\\
& u^i_{\{k,j\}}\in\mathbb{R}_+, && \forall \Lambda^i_{\{k,j\}}\subset E. &&
\end{aligned}
\tag{8.18}
$$

Objective function z_- (8.18.0): the energy consumption of data transmissions per unit of time, after removing the energy cost saved by network coding.

Flow conservation constraints (8.18.1) to (8.18.3): incoming and outgoing flows at each node are balanced.

Incident edge flow constraints (8.18.4): the flow q_{jik}^{st} on incident edges (j,i) and (i,k) is equal to d^{st} if the demand st uses an unsplittable path through (j,i) and (i,k), otherwise $q_{jik}^{st}=0$.

Coding opportunity constraints (8.18.5) and (8.18.6): the coding variable $u^i_{\{k,j\}}$ (cf. inequality (8.7)) is at most the minimum of two opposite aggregated flows along $j\to i\to k$ and $k\to i\to j$.

Clique capacity constraint (8.18.7) with network coding: the OTR within the clique set m should be at most 1 (cf. (8.14)).

The wUMCFC-EQ formulation contains the classical constraints of the UMCF problem, and additional clique capacity constraints, quadratic constraints and network coding constraints and variables. The next section is dedicated to presenting an edge linearization of the wUMCFC-EQ problem.

Edge linearization formulation

We propose an MILP reformulation of the nonlinear MBQP formulation.

The constraint $q_{jik}^{st} = d^{st}x_{ji}^{st}x_{ik}^{st}$ is quadratic, x_{ji}^{st} and x_{ik}^{st} are binary variables, so we propose the exact linearization approach as follows:

$$
\begin{aligned}
& q_{jik}^{st} \ge d^{st}(1 - x_{ji}^{st} - x_{ik}^{st}),\\
& q_{jik}^{st} \le d^{st}x_{ji}^{st},\\
& q_{jik}^{st} \le d^{st}x_{ik}^{st},\\
& q_{jik}^{st} \ge 0.
\end{aligned}
\tag{8.19}
$$

The edge linearization formulation, denoted by wUMCFC-EL, replaces incident edge flow constraints (8.18.4) of the wUMCFC-EQ formulation with the following linearization constraints:

$$q^{st}_{jik} \geq d^{st}(1 - x^{st}_{ji} - x^{st}_{ik}), \qquad \forall (j,i) \in E,\ \forall (i,k) \in E,\ \forall st \in D, \tag{8.20}$$

$$q^{st}_{jik} \leq d^{st} x^{st}_{ji}, \qquad \forall (j,i) \in E,\ \forall (i,k) \in E,\ \forall st \in D, \tag{8.21}$$

$$q^{st}_{jik} \leq d^{st} x^{st}_{ik}, \qquad \forall (j,i) \in E,\ \forall (i,k) \in E,\ \forall st \in D, \tag{8.22}$$

$$q^{st}_{jik} \geq 0, \qquad \forall (j,i) \in E,\ \forall (i,k) \in E,\ \forall st \in D. \tag{8.23}$$

Indeed, the wUMCFC-EL is a standard linearization of the wUMCFC-EQ problem, which is currently used by commercial solvers such as CPLEX [70].

Edge balance formulation

The edge balance formulation uses the balanced property of flows on edges to represent the flows on incident edges. We denote the edge balance formulation by wUMCFC-EB. More precisely, the wUMCFC-EB formulation replaces incident edge flow constraints (8.18.4) of the wUMCFC-EQ formulation with the following edge balance constraints:

$$\begin{aligned} \sum_{(i,k)\in E} q^{st}_{jik} - d^{st} x^{st}_{ji} = 0,\ \ \forall (j,i) \in E,\ \ \forall st \in D, \\ \sum_{(j,i)\in E} q^{st}_{jik} - d^{st} x^{st}_{ik} = 0,\ \ \forall (i,k) \in E,\ \ \forall st \in D. \end{aligned} \tag{8.24}$$

The wUMCFC-EB is an MILP formulation.

We illustrate the usage of subscript/superscript notation for the subsequent part of this chapter. When we omit subscripts and/or superscripts of some variables, we denote the subset of variables restricted to the remaining subscripts and/or superscripts. For example, x denotes the set of flow variables indexed by entire superscripts (demands) and subscripts (edges), and x^{st} denotes the set of $st-$flow variables indexed by entire subscripts (edges).

The wUMCFC-EB is a reformulation of the wUMCFC-EQ formulation by the following theorem.

Theorem 8.1. *Let x take binary values satisfying flow conservation constraints on nodes (8.18.1), (8.18.2) and (8.18.3). Then q satisfies (8.18.4) if and only if q satisfies* (8.24).

Proof. Assume x takes binary values satisfying flow conservation constraints on nodes (8.18.1), (8.18.2) and (8.18.3). Then, x already represents an UMCF. For any (j,i) and $(i,k) \in E$ and $st \in D$, $q^{st}_{jik} = d^{st} x^{st}_{ji} x^{st}_{ik}$ if and only if q^{st}_{jik} is the value of binary st-flow over incident edges (j,i) and (i,k). For each $st \in D$, since x^{st} is an unsplittable flow, the $st-$flow takes a unique path from s to t. Then the latter condition is equivalent to: for any edge $(i,j) \in E$, the $st-$ flow entering (i,j) from its incident edges equals the $st-$ flow leaving from (i,j) to its incident edges, which is exactly the edge balance constraint (8.24). □

According to the following theorem, the linear relaxation of the wUMCFC-EL formulation is not stronger than the linear relaxation of the wUMCFC-EB formulation.

Theorem 8.2. *The optimal value of the linear relaxation of the wUMCFC-EL formulation is at most the optimal value of the wUMCFC-EB formulation.*

Proof. Let $(\bar{x}, \bar{q}, \bar{u})$ be a feasible solution of the linear relaxation of the wUMCFC-EB formulation. Since the $\bar{x}$ of the relaxation represents a fractional MCF, for each $st \in D$, we can partition $\bar{x}^{st}$ and d^{st} into a finite number of unsplittable $st-$flows. Denote by χ^{st} the set of paths on which $st-$flows in the relaxation solution have non-zero values, and let $I_{ij}^{p} \in \{0,1\}$ ($(i,j) \in E$) be the Boolean indicating whether a path p contains the edge (i,j), and let $d^{st,p}$ be the value of the $st-$flow routed by path p. It follows that

$$\begin{aligned} d^{st}\bar{x}_{ij}^{st} &= \sum_{p \in \chi^{st}} d^{st,p} I_{ij}^{p}, \\ d^{st} &= \sum_{p \in \chi^{st}} d^{st,p} \end{aligned} \tag{8.25}$$

According to the edge balance constraints (8.24), $\bar{q}_{jik}^{st}$ ((j,i) and $(i,k) \in E$) could be decomposed by unsplittable $st-$flows in χ^{st}. Let $\bar{q}_{jik}^{st,p}$ be the $st-$flow of path p containing edge (j,i) and (i,k); if p does not contain edge (j,i) and (i,k), $\bar{q}_{jik}^{st,p}$ is defined as zero. It follows that

$$\begin{aligned} \bar{q}_{jik}^{st} &= \sum_{p \in \chi^{st}} \bar{q}_{jik}^{st,p}, \\ \frac{\bar{q}_{jik}^{st,p}}{d^{st,p}} &\in \{0,1\}, \\ \frac{\bar{q}_{jik}^{st,p}}{d^{st,p}} &= I_{ji}^{p} I_{ik}^{p}. \end{aligned} \tag{8.26}$$

The reformulation of $\frac{\bar{q}_{jik}^{st,p}}{d^{st,p}} = I_{ji}^{p} I_{ik}^{p}$ is obtained by using the following inequalities:

$$\begin{aligned} \bar{q}_{jik}^{st,p} &\geq d^{st,p}(1 - I_{ji}^{p} - I_{ik}^{p}), \\ \bar{q}_{jik}^{st,p} &\leq d^{st,p} I_{ji}^{p}, \\ \bar{q}_{jik}^{st,p} &\leq d^{st,p} I_{ik}^{p}, \\ \bar{q}_{jik}^{st,p} &\geq 0. \end{aligned} \tag{8.27}$$

By summing up these four inequalities over $p \in \chi^{st}$, and substituting equations (8.25), we obtain:

$$\begin{aligned} \bar{q}_{jik}^{st} &\geq d^{st}(1 - \bar{x}_{ji}^{st} - \bar{x}_{ik}^{st}), \\ \bar{q}_{jik}^{st} &\leq d^{st}\bar{x}_{ji}^{st}, \\ q_{jik}^{st} &\leq d^{st}\bar{x}_{ik}^{st}, \\ q_{jik}^{st} &\geq 0, \end{aligned} \tag{8.28}$$

which are exactly constraints (8.19) of the wUMCFC-EL formulation. Therefore, $(\bar{x}, \bar{q}, \bar{u})$ is also a feasible solution of the linear relaxation of the wUMCFC-EL formulation. The feasible set of the linear relaxation of the wUMCFC-EL formulation includes the feasible set of the wUMCFC-EB formulation so the result follows. □

The wUMCFC-EL formulation is not a stronger formulation, and the linearization (8.20) introduces more constraints compared to (8.24). The wUMCFC-EB formulation is more suitable from the computational point of view, so we only solve wUMCFC-EB (8.18) in our experiments (see Section 8.6).

The edge-based formulations cannot be applied to large-scale problems because their number of variables and constraints increase dramatically with the size of the network and the number of demands. In the next section, we propose a new formulation using path variables which leads to a better modelisation of the unsplittable routing and network coding.

8.3.2 Path-based formulation

We next consider the approach in Sect. 3.1.1. The path-based formulation is a DW reformulation of the wUMCFC-EB formulation, where path variables are extreme points of the convex hull of edge variables. Their linear relaxations have the same value, but the sizes of LPs and the times to solve corresponding MILPs are different (see Section 8.6). The path-based formulation contains an exponential number of path variables w.r.t. the size of the graph, but it can be solved efficiently by the column generation approaches.

Denote by φ^{st} the set of all simple paths from the source s to the target t. We define the following decision variables used by the path-based formulation:

Decision variables:

y_p^{st} : binary variable indicating whether demand $st \in D$ is routed along a path $p \in \varphi^{st}$.

$u^i_{\{k,j\}}$: real coding variable denoting the amounts of two opposite flows, along $k \to i \to j$ and $j \to i \to k$, which can be encoded at i.

The path-based MILP formulation, denoted by wUMCFC-P, follows as:

$$\min \quad z_- = \sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(i,j)\in p}\beta_{ij}d^{st}y_p^{st} - \sum_{\Lambda^i_{\{k,j\}}\subset E}\tau^i_{\{k,j\}}u^i_{\{k,j\}} \tag{8.29.0}$$

$$\begin{aligned}
& u^i_{\{k,j\}} - \sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(k,i),(i,j)\in p} d^{st}y_p^{st} \le 0, && && \\
& u^i_{\{k,j\}} - \sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(j,i),(i,k)\in p} d^{st}y_p^{st} \le 0, && \forall \Lambda^i_{\{k,j\}} \subset E, && (8.29.1)\\
& \sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(i,j)\in p\cap m}\frac{d^{st}}{C_{ij}}y_p^{st} - \sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{u^i_{\{k,j\}}}{C^i_{\{k,j\}}} \le 1, && \forall m \in M, && (8.29.2)\\
& \sum_{p\in\varphi^{st}} y_p^{st} = 1, && \forall st \in D, && (8.29.3)\\
& y_p^{st} \in \{0,1\}, && \forall st \in D,\ \forall p \in \varphi^{st}, && \\
& u^i_{\{k,j\}} \in \mathbb{R}_+, && \forall \Lambda^i_{\{k,j\}} \subset E &&
\end{aligned} \tag{8.29}$$

Objective function z_- (8.29.0): energy consumption of data transmissions in the network per unit of time.

Coding opportunity constraints (8.29.1): every three-node coding opportunity set induces a pair of constraints, associated with two opposite flows, such that the coding variable $u^i_{\{k,j\}}$ (cf. inequality (8.7)) is at most the aggregated flows along $k \to i \to j$ and $j \to i \to k$.

Clique capacity constraints (8.29.2) with network coding: the OTR within the clique set m should be at most 1 (cf. (8.14)).

Unsplittable constraints (8.29.3): each demand has to be routed by a single path.

The path-based formulation wUMCFC-P has an exponential number of path variables, but it contains fewer other variables and fewer constraints than the edge-based formulations. The numerical experiments in Section 8.6 show that it can be solved efficiently by the branch-and-price algorithm.

Table 8.1 Comparison of variables

	Routing vars	Coding vars	Auxiliary vars
Path-based formulation	Exponential	$O(\|E\|^2)$	0
Edge-based formulation	$O(\|E\|\|D\|)$	$O(\|E\|^2)$	$O(\|E\|^2\|D\|)$

	#Flow cons	#Coding opportunity cons	#Clique capacity cons
Path-based formulation	$O(\|D\|)$	$O(\|E\|^2)$	$O(\|M\|)$
Edge-based formulation	$O(\|D\|\|V\|)$	$O(\|E\|\|D\|)$ + $O(\|E\|^2)$	$O(\|M\|)$

Table 8.2 Comparison of constraints

Tables (8.1) and (8.2) compare sizes of edge and path-based formulations in terms of number of variables and number of constraints.

8.4 Column generation

This section is dedicated to presenting a column generation approach to solve the LP relaxation of wUMCFC-P (8.29).

The column generation approaches [124, 152, 215, 295] start by solving an initial linear program (LP) restricted to a small subset of variables and then generate new variables (columns) dynamically. Columns are removed from the LP because there are too many columns to handle efficiently, and most of them will have their associated variable equal to zero in an optimal solution. The LP problem with all columns is called the master problem, and the problem limited to a subset of active columns is called the restricted master problem (RMP). To check the optimality of the RMP solution, a subproblem, called the pricing problem, is solved to try to identify columns with a negative reduced cost (for a minimization problem). If such columns are found, the RMP is reoptimized. When the pricing subproblem returns a solution with a nonnegative reduced cost, we can conclude that the solution to the master problem is optimal.

The pricing algorithm developed here consists of two parts. The first part is called reduced cost pricing, and it adds paths with the minimum negative reduced cost. The second part is called Farkas pricing which identifies and repairs the infeasibility. We reduce the pricing problem to a shortest path problem in an extended graph.

8.4.1 Reduced cost pricing

The objective function of the pricing subproblem is the reduced cost of the new variable with respect to the current dual variables. From the solution of the RMP, we obtain the dual prices for each of the constraints in the RMP. This information is then used in the objective function of the pricing problem.

If the objective value of the pricing problem is negative, a variable with negative reduced cost is identified. This variable is then added to the RMP, and the RMP is re-optimized. Re-solving the RMP will generate a new set of dual values, and the process is repeated until no negative reduced cost variables are identified.

Let $\gamma_{jik}, \gamma_{kij}, \zeta_m$ and η_{st} be the dual variables associated respectively to the constraints (8.29.1), (8.29.2) and (8.29.3).

The reduced cost of a path variable on $p \in \varphi^{st}$ is denoted by $\mathrm{RC}^{st}(p)$ and

$$\mathrm{RC}^{st}(p) \coloneqq \sum_{(k,i),(i,j)\in p} -d^{st}\gamma_{kij} + \sum_{(i,j)\in p} d^{st}\beta_{ij} + \sum_{m\in M}\sum_{(i,j)\in p\cap m} d^{st}\frac{\zeta_m}{C_{ij}} + \eta_{st}. \tag{8.30}$$

The dual problem of the LP relaxation of wUMCFC-P (Dual-wUMCFC-P) follows as:

$$\begin{aligned}
\max \quad & -\sum_{m\in M}\zeta_m - \sum_{st\in D}\eta_{st} && && (8.31.0)\\
& \gamma_{kij} + \gamma_{jik} - \tau^i_{\{k,j\}} - \sum_{m\in M:\Lambda^i_{\{k,j\}}\subset m}\frac{\zeta_m}{C^i_{\{k,j\}}} = 0, && \forall \Lambda^i_{\{k,j\}} \subset E, && (8.31.1)\\
& \mathrm{RC}^{st}(p) \geq 0, && \forall st \in D, \forall p \in \varphi^{st}, && (8.31.2)\\
& \zeta_m \in \mathbb{R}_+, && \forall m \in M, && (8.31.3)\\
& \gamma_{kij}, \gamma_{jik} \in \mathbb{R}_+, && \forall \Lambda^i_{\{k,j\}} \subset E, && (8.31.4)\\
& \eta_{st} \in \mathbb{R}, && \forall st \in D. && (8.31.5)
\end{aligned} \tag{8.31}$$

The dual variables γ domain is originally defined over the opportunity sets $\Lambda \subset E \times E$. We extend this domain to $E \times E$ so the subsequent analysis is simpler: for $(k,i),(i,j),(j,i),(i,k) \in E$, if $\Lambda^i_{\{k,j\}} \not\subset E$, we define $\gamma_{kij} = 0$ and $\gamma_{jik} = 0$.

Let $\sigma^{st} \subset \varphi^{st}$ ($st \in D$) be the set of active path variables added into RMP; only constraints (8.31.2) indexed by σ^{st} are included in the dual RMP.

The pricing problem is decomposed into $|D|$ sub-problems. For each $st \in D$, the corresponding sub-problem checks whether there exists a path p from s to t such that $\mathrm{RC}^{st}(p) < 0$.

We define a path cost function PC, such that for a path p in the graph G we have:

$$\mathrm{PC}(p) \coloneqq \sum_{(k,i),(i,j)\in p} -\gamma_{kij} + \sum_{(i,j)\in p}\beta_{ij} + \sum_{m\in M}\sum_{(i,j)\in p\cap m}\frac{\zeta_m}{C_{ij}}. \tag{8.32}$$

Let $p^* = \mathrm{argmin}_{p\in\varphi^{st}}\,\mathrm{PC}(p)$, it follows that $\mathrm{RC}^{st}(p^*) = d^{st}\mathrm{PC}(p^*) + \eta_{st} = \min_{p\in\varphi^{st}}\mathrm{RC}^{st}(p)$. If $\mathrm{RC}^{st}(p^*) < 0$, then we add the column associated to the path p^* to the RMP.

The pricing problem is reduced to finding a path p with the minimum cost $\mathrm{PC}(p)$.

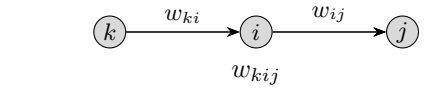
The shortest path algorithm is not used in the original graph G since $\mathrm{PC}(p)$ includes positive and negative costs on incident edges. However, we have the following polynomial reduction of the pricing problem to the shortest path problem in an extended graph.

Extended Graph

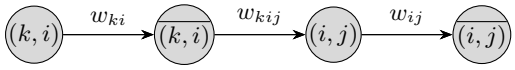
Denote by $EG = (H, A)$ the weighted directed extended graph to construct, and let w be the weight function over edges in A. Let $\overline{E} = \left\{\overline{(i,j)} \,|\, (i,j) \in E\right\}$ be auxiliary edges that are copies of original edges.

We build EG as the follows:

1. Let $H \coloneqq E \cup \overline{E}$.



(a) Edges in original graph G



(b) Corresponding nodes in extended graph EG

Figure 8.3 G and EG

2. If $(k,i) \in E$ and $(i,j) \in E$, then $\left(\overline{(k,i)}, (i,j)\right) \in A$;
   If $(i,j) \in E$, then $\left((i,j), \overline{(i,j)}\right) \in A$.
   Denote

$$\begin{aligned} A^{\leq} &:= \left\{ \left((\overline{(k,i)}, (i,j) \right) \mid (k,i), (i,j) \in E \right\}, \\ A^{\geq} &:= \left\{ \left((i,j), \overline{(i,j)} \right) \mid (i,j) \in E \right\}. \end{aligned} \tag{8.33}$$

   Consequently, $A = A^{\leq} \cup A^{\geq}$ is partitioned into two subsets.

3. For $\left(\overline{(k,i)}, (i,j)\right) \in A^{\leq}$, its weight is

$$w\left(\left(\overline{(k,i)}, (i,j)\right)\right) := -\gamma_{kij}, \tag{8.34}$$

   and we abbreviate it as w_{kij}, which is negative. We call $A^{\leq}$ the set of negative edges.
   For $\left((i,j), \overline{(i,j)}\right) \in A^{\geq}$, its weight is

$$w\left(\left((i,j), \overline{(i,j)}\right)\right) := \beta_{ij} + \sum_{m \in M : (i,j) \in m} \frac{\zeta_m}{C_{ij}}, \tag{8.35}$$

   and we abbreviate it as w_{ij}, which is positive. We call $A^{\geq}$ the set of positive edges.

   By construction, negative edges are only incident to positive edges, and vice versa.

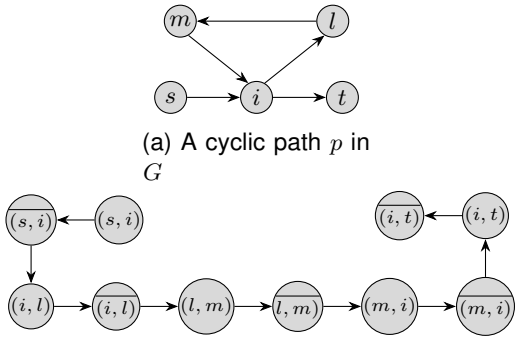
Figure (8.3) is an example of building an extended graph.

The size of the extended graph EG is polynomial in the size of G: EG has $O(|E|)$ nodes and $O(|E|^2)$ edges.

The weight $w(p')$ on a path p' in EG is defined as the sum of its edge weights. Two paths representations are used: the edge representation, which is an ordered sequence of edges enclosed by $()$, and the node representation as an ordered sequence of nodes enclosed by $[]$. The first (resp. last) node of a path is called the source (resp. target) node. We define the following mapping:

Definition 8.3. *Let a path p in G be $(e_1, \ldots, e_n)$ where $e_i \in E$ for $i = 1, \ldots, n$, and denote by $\overline{e} = \overline{(i,j)}$ for $e = (i,j) \in E$. The path mapping π maps p to a path $\pi(p)$ in EG satisfying:* $\pi(p) = [e_1, \overline{e_1}, \ldots, e_n, \overline{e_n}]$.

Figure (8.3) shows that π maps a path $((k,i),(i,j))$ in (8.3a) to a path $\left[(k,i), \overline{(k,i)}, (i,j), \overline{(i,j)}\right]$ in (8.3b).



(a) A cyclic path p in G

(b) A simple path $p' = \pi(p)$ in EG

Figure 8.4 π^{-1} does not preserve the acyclicity from EG to G

Therefore, the source and target of p are s and t if and only if the source and target of $\pi(p)$ are (s, i) and $\overline{(j, t)}$ for some $i \in V$ and some $j \in V$.

φ^{st} is the set of simple paths in G from node $s \in V$ to node $t \in V$.

We define ψ^{st} as the set of simple paths in EG in which source nodes are $(s, i) \in E$ for some $i \in V$ and target nodes are $\overline{(j, t)}$ for some $j \in V$.

Lemma 8.4. *Given $st \in D$, the following properties are satisfied by the mapping π:*

1. *π is an injection.*
2. *For $p \in \varphi^{st}$, $PC(p) = w(\pi(p))$, where w is the sum of weights of edges of $\pi(p)$.*
3. *$\pi(\varphi^{st}) \subsetneq \psi^{st}$.*

Proof. From the construction of EG and π: π is injective, $\mathrm{PC}(p) = w(\pi(p))$ and $\pi(\varphi^{st}) \subset \psi^{st}$. To prove the strictness of the inclusion, we give an example in Figure (8.4).

$$p = ((s, i), (i, l), (l, m), (m, i), (i, t))$$

is cyclic,

$$p' = \pi(p) = \left[(s, i), \overline{(s, i)}, (i, l), \overline{(i, l)}, (l, m), \overline{(l, m)}, (m, i), \overline{(m, i)}, (i, t), \overline{(i, t)}\right]$$

is simple, thus $\pi(\varphi^{st}) \neq \psi^{st}$. □

Every path p' in EG with source node in E and target node in $\overline{E}$ is the map of a unique path p in G (see Figure (8.4)), and the inverse π^{-1} is well-defined on those p'. Since π preserves the path cost and is an injection, the pricing problem is reduced to finding $\mathrm{argmin}_{p' \in \pi(\varphi^{st})} w(p')$, and recovering the inverse path.

There are two issues to address:

- Enumerating all paths in $\pi(\varphi^{st})$ is not efficient. Note that ψ^{st} is a set of paths in EG with known sources in E and targets in $\overline{E}$, but it strictly includes $\pi(\varphi^{st})$.
  Let $p^* = \mathrm{argmin}_{p' \in \psi^{st}} w(p')$ be a shortest path in ψ^{st}; it follows that

$$w(p^*) \leq \min_{p' \in \pi(\varphi^{st})} w(p') = \min_{p \in \varphi^{st}} \mathrm{PC}(p). \tag{8.36}$$

We will prove that $p^* \in \pi(\varphi^{st})$ (equivalently, $\pi^{-1}(p^*)$ is simple), and hence $w(p^*) = \min_{p' \in \pi(\varphi^{st})} w(p')$.

- The path p^* is a shortest simple path in EG between a known set of sources and targets indicated by ψ^{st}. Note that there exist edges with negative weights in EG. We will prove that none of cycles in EG are negative, i.e., EG is conservative. Hence, the shortest simple path problem is solvable in polynomial time.

The negative weights are adressed first.

Lemma 8.5. *For* $\Lambda^i_{\{k,j\}} \subset E$, $w_{ik} + w_{kij} + w_{jik} \geq 0$ *and* $w_{ij} + w_{kij} + w_{jik} \geq 0$.

Proof. From the constraint (8.31.1) and $\frac{1}{C^i_{\{k,j\}}} \leq \min\left\{\frac{1}{C_{ij}}, \frac{1}{C_{ik}}\right\}$ (cf. (8.12)), we have

$$\gamma_{kij} + \gamma_{jik} - \tau^i_{\{k,j\}} = \sum_{m \in M : \Lambda^i_{\{k,j\}} \subset m} \frac{\zeta_m}{C^i_{\{k,j\}}} \leq \sum_{m \in M : \Lambda^i_{\{k,j\}} \subset m} \frac{\zeta_m}{C_{ij}} \leq \sum_{m \in M : (i,j) \in m} \frac{\zeta_m}{C_{ij}}. \quad (8.37)$$

It follows from the definition of w_{kij}, w_{jik} and w_{ij} that

$$\begin{aligned} & - w_{kij} - w_{jik} \\ =& \gamma_{kij} + \gamma_{jik} \\ \leq& \tau^i_{\{k,j\}} + \sum_{m \in M : (i,j) \in m} \frac{\zeta_m}{C_{ij}} \\ \leq& \beta_{ij} + \sum_{m \in M : (i,j) \in m} \frac{\zeta_m}{C_{ij}} \\ =& w_{ij}. \end{aligned} \quad (8.38)$$

The last inequality follows from (8.17).

Hence, $w_{ij} + w_{kij} + w_{jik} \geq 0$ holds. The proof for $w_{ik} + w_{kij} + w_{jik} \geq 0$ is similar. □

Theorem 8.6. *Let* e_1 *and* e_2 *be two incident edges of* EG*, let* e_1 *be negative, and let* e_2 *be positive. Then,* $w(e_1) + w(e_2) \geq 0$

Proof. Let $e_1 = \left(\overline{(k,i)}, (i,j)\right)$ be negative and $e_2 = \left((i,j), \overline{(i,j)}\right)$ be positive. If $\Lambda^i_{\{k,j\}} \subset E$, it follows from Lemma (8.5) that

$$\begin{aligned} & w(e_1) + w(e_2) \\ =& w_{kij} + w_{ij} \\ \geq& - w_{jik} \\ \geq& 0. \end{aligned} \quad (8.39)$$

Otherwise, according to the extension of γ in Subsection 8.4.1 and the definition of weights, $w(e_1) = w_{kij} = 0$ and $w(e_2) = w_{ij} \geq 0$, so $w(e_1) + w(e_2) \geq 0$. □

Corollary 8.7. *For any simple cycle* r *of* EG*, it follows that* $w(r) \geq 0$.

Proof. Denote $r = (e_1, ..., e_t)$, and let its length be t. Because of alternative appearances of positive and negative edges in the cycle, t is even where $t = 2h$ for some integer h. W.l.o.g., we assume e_1 is negative.

By Lemma (8.6), then $w(r) = \sum_{h\in\{1,\dots,\frac{t}{2}\}} (w(e_{2h-1}) + w(e_{2h})) \geq 0$. □

The Bellman-Ford algorithm is used to compute the shortest simple path p^* in the graph EG. The number of nodes and edges of the extended graph are $O(|E|)$ and $O(|E|^2)$, respectively, hence the time complexity is $O(|E|^3)$.

The complexity of the pricing algorithm is $O(|D||E|^3)$. The Bellman-Ford algorithm can output a smallest length path when there are multiple shortest paths of the same weight, so the pricing algorithm works even if the weight of a cycle is zero. Lemma 8.8 demonstrates that the removal of cycles of flows would decrease the objective value.

Now, we prove that $\pi^{-1}(p^*)$ is simple based on the perturbation analysis and the optimality of p^*.

Let (y, u) be a feasible solution of LP relaxation of (8.31), and denote by $z_-(y, u)$ the objective value of solution (y, u). If there exists a path p in G such that $y_p > 0$, and p contains a cycle r, we call this solution is cyclic.

Here, we take the edge representation of a path/cycle, and $p \setminus r$ denotes the usual set minus operation. Since the left-hand sides of clique capacity constraints and the objective function are related to OTRs, deleting cycles of a path would reduce the OTRs on the edges of cycles. As a result, deleting cycles could both decrease the objective and increase the feasibility.

Lemma 8.8. *Let $(\bar{y}, \bar{u})$ be a cyclic solution such that there exists a path $\dot{p}$ with $\bar{y}_{\dot{p}} > 0$, and $\dot{p}$ contains a cycle $\dot{r}$; denote by $\ddot{p} = \dot{p} \setminus \dot{r}$ a path without the cycle $\dot{r}$. There exists another feasible solution $(\widehat{y}, \widehat{u})$, such that $\widehat{y}_{\dot{p}} = 0$, $\widehat{y}_{\ddot{p}} = \bar{y}_{\dot{p}} + \bar{y}_{\ddot{p}}$, $\widehat{y}_p = \bar{y}_p$ for $p \neq \dot{p}$ and $\ddot{p}$, and $z_-(\widehat{y}, \widehat{u}) \leq z_-(\bar{y}, \bar{u})$.*

Proof. We can assume that $\widehat{u}$ (resp. $\bar{u}$) is equal to its upper-bound defined by the pair of constraints (8.31.1) for fixed $\widehat{y}$ (resp. $\bar{y}$). This is because increasing u does not change the feasibility of the solution and decreases the objective value.

Assume $\bar{y}$ follows the value assignment of the lemma, and $\bar{u}$ is equal to its upper-bound defined by constraints (8.31.1) with fixed $\bar{y}$.

In contrast to the path $\dot{p}$, the path $\ddot{p}$ does not contain the cycle $\dot{r}$. Denote by $\dot{r}^+ = \dot{r} \cup \{(j, i) \,|\, (i, j) \in \dot{r}\}$ the union of the cycle $\dot{r}$ and its reversed cycle. For $\Lambda^i_{\{k,j\}} \subset \dot{r}^+ \cap E$, the flow on cycle $\dot{r}$ affects the coding variable $\widehat{u}^i_{\{k,j\}}$. By deleting the cycle $\dot{r}$, the right-hand side of one of the constraint pairs (8.31.1) decreases by at most $\bar{y}_{\dot{p}}$.

Coding variables affected by cycle $\dot{r}$ satisfy the following conditions

$$\bar{u}^i_{\{k,j\}} - \widehat{u}^i_{\{k,j\}} \leq d^{st}\bar{y}_{\dot{p}}, \qquad \forall \Lambda^i_{\{k,j\}} \subset \dot{r}^+ \cap E, \tag{8.40}$$

$$\bar{u}^i_{\{k,j\}} - \widehat{u}^i_{\{k,j\}} = 0, \qquad \forall \Lambda^i_{\{k,j\}} \not\subset \dot{r}^+ \cap E. \tag{8.41}$$

We first check that the clique capacity constraint (8.31.2) is still feasible for each clique set $m \in M$.

Assume that $\dot{p} \in \varphi^{st}$. If $r \cap m = \emptyset$, then $\dot{p} \cap m = \emptyset$. Hence, $\widehat{y}_p = \bar{y}_p$ for p s.t. $p \cap m \neq \emptyset$, and $\bar{u}^i_{\{k,j\}} = \widehat{u}^i_{\{k,j\}}$ for all $\Lambda^i_{\{k,j\}} \subset m$. So the left-hand side of the clique capacity constraints on m remains the same (feasible). Otherwise if $r \cap m \neq \emptyset$, then for $\Lambda^i_{\{k,j\}} \subset m \cap \dot{r}^+$, the associated coding variables decrease from $\bar{u}^i_{\{k,j\}}$ to $\widehat{u}^i_{\{k,j\}}$.

It follows that

$$\begin{aligned}
&\sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{1}{C^i_{\{k,j\}}}\bar{u}^i_{\{k,j\}}-\sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{1}{C^i_{\{k,j\}}}\widehat{u}^i_{\{k,j\}}\\
=&\sum_{\Lambda^i_{\{k,j\}}\subset m\cap\dot{r}^+}\frac{1}{C^i_{\{k,j\}}}\bar{u}^i_{\{k,j\}}-\sum_{\Lambda^i_{\{k,j\}}\subset m\cap\dot{r}^+}\frac{1}{C^i_{\{k,j\}}}\widehat{u}^i_{\{k,j\}}\\
\leq&\sum_{\Lambda^i_{\{k,j\}}\subset m\cap\dot{r}^+}\frac{d^{st}}{C^i_{\{k,j\}}}\bar{y}_{\dot{p}}\\
\leq&\sum_{\Lambda^i_{\{k,j\}}\subset m\cap\dot{r}^+}\min\left\{\frac{d^{st}}{C_{ij}},\frac{d^{st}}{C_{ik}}\right\}\bar{y}_{\dot{p}}\\
\leq&\sum_{(i,j)\subset m\cap r}\frac{d^{st}}{C_{ij}}\bar{y}_{\dot{p}},
\end{aligned}\tag{8.42}$$

the second inequality follows from (8.12) and the third inequality follows from an enlarged index set.

We check the left-hand side of the clique capacity constraint for each m:

$$\begin{aligned}
&\sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(i,j)\in p\cap m}\frac{d^{st}}{C_{ij}}\widehat{y}_p-\sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{1}{C^i_{\{k,j\}}}\widehat{u}^i_{\{k,j\}}\\
=&\sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(i,j)\in p\cap m}\frac{d^{st}}{C_{ij}}\bar{y}_p-\sum_{(i,j)\in m\cap r}\frac{d^{st}}{C_{ij}}\bar{y}_{\dot{p}}-\sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{1}{C^i_{\{k,j\}}}\widehat{u}^i_{\{k,j\}}\\
\leq&\sum_{st\in D}\sum_{p\in\varphi^{st}}\sum_{(i,j)\in p\cap m}\frac{d^{st}}{C_{ij}}\bar{y}_p-\sum_{\Lambda^i_{\{k,j\}}\subset m}\frac{1}{C^i_{\{k,j\}}}\bar{u}^i_{\{k,j\}}\\
\leq&1.
\end{aligned}\tag{8.43}$$

The first equation follows from the difference between $\bar{y}$ and $\widehat{y}$ on the cycle $\dot{r}$, the first inequality follows from (8.42), and the last inequality follows from the fact that $\bar{y}$ is feasible. Indeed, the left-hand side of the clique capacity constraint under the new solution $(\widehat{y},\widehat{u})$ is decreased.

Since the $st-$flow on cycle $\dot{r}$ decreases by $\bar{y}_{\dot{p}}$, and coding variables decrease accordingly, it follows that:

$$\begin{aligned}
&z_-(\widehat{y},\widehat{u})-z_-(\bar{y},\bar{u})\\
=&-\sum_{(i,j)\in\dot{r}}\beta_{ij}d^{st}\bar{y}_{\dot{p}}-\sum_{\Lambda^i_{\{k,j\}}\subset\dot{r}^+\cap E}\tau^i_{\{k,j\}}\left(\widehat{u}^i_{\{k,j\}}-\bar{u}^i_{\{k,j\}}\right)\\
\leq&-\sum_{(i,j)\in\dot{r}}\beta_{ij}d^{st}\bar{y}_{\dot{p}}+\sum_{\Lambda^i_{\{k,j\}}\subset\dot{r}^+\cap E}\tau^i_{\{k,j\}}d^{st}\bar{y}_{\dot{p}}\\
=&-\sum_{(i,j)\in\dot{r}:\nexists k,\Lambda^i_{\{k,j\}}\subset\dot{r}^+\cap E}\beta_{ij}d^{st}\bar{y}_{\dot{p}}+\sum_{(i,j)\in\dot{r}:\exists k,\Lambda^i_{\{k,j\}}\subset\dot{r}^+\cap E}(-\beta_{ij}+\tau^i_{\{k,j\}})d^{st}\bar{y}_{\dot{p}}\\
\leq&-\sum_{(i,j)\in\dot{r}:\nexists k,\Lambda^i_{\{k,j\}}\subset\dot{r}^+\cap E}\beta_{ij}d^{st}\bar{y}_{\dot{p}}\\
\leq&0.
\end{aligned}\tag{8.44}$$

The first equation follows from the difference of objective values on cycle $\dot{r}$, the first inequality follows from (8.40), and the second equation follows from the partition of edges in $\dot{r}$, and the second inequality follows from (8.17).

Therefore, $z_-(\widehat{y}, \widehat{u}) \leq z_-(\bar{y}, \bar{u})$. □

Recall that $p^* = \operatorname{argmin}_{p' \in \psi^{st}} w(p')$, and its path cost $\mathrm{PC}(\pi^{-1}(p^*)) = \min_{p' \in \psi^{st}} w(p') \leq \min_{p' \in \varphi^{st}} w(p')$. We further conclude the following theorem 8.9.

Theorem 8.9. *$\pi^{-1}(p^*)$ is a simple path in G, and $PC(\pi^{-1}(p^*)) = \min_{p' \in \varphi^{st}} w(p')$.*

Proof. Note that the reduced cost $\mathrm{RC}^{st}(\pi^{-1}(p^*))$ of p^* is the minimum among paths in ψ^{st}. By the definition of the minimum reduced cost, among all path variables which admit zero values in RMP, assume a positive perturbation of the path variable associated to $\pi^{-1}(p^*)$, such perturbation decreases the objective function.

If $\pi^{-1}(p^*)$ is not simple, by Lemma (8.8), there are two cases:

- If deleting the cycles of p^* decreases the objective value, it would contradict the minimum reduced cost of $\pi^{-1}(p^*)$;
- Otherwise, deleting cycles yields a solution of the same objective value, i.e., the reduced cost of the new simple path is the same as the reduced cost of $\pi^{-1}(p^*)$. But according to the Bellman-Ford algorithm, among paths of the same minimum cost, only the shortest simple path would be the output of the algorithm, which contradicts our assumption.

Therefore, $\pi^{-1}(p^*)$ must be simple and the result follows. □

Therefore, there is no need to check whether an inverse optimal path is simple. To summarize, the pricing algorithm constructs the extended graph EG, for each $st \in D$, finds a shortest path in φ^{st} (Bellman-Ford Algorithm), and adds the inverse path in σ^{st}. If no such path exists, the master problem is optimal.

8.4.2 Farkas pricing

The Farkas pricing improves the feasibility when the LP relaxation of the RMP is infeasible.

The constraint (8.31.2) differentiates the RMP dual problem from the master dual problem. The quantifier φ^{st} in Dual- wUMCFC-P (8.31) is replaced by its active subset σ^{st}.

According to the Farkas' lemma, the RMP is infeasible if and only if the dual RMP is unbounded. The dual RMP is unbounded if and only if there exists an improving ray $(\Delta(\eta), \Delta(\gamma), \Delta(\zeta))$ (Farkas certificate) of the dual RMP along which the objective function can be improved infinitely.

The dual simplex algorithm could detect this improving ray. The vector $(\Delta(\eta), \Delta(\gamma), \Delta(\zeta))$ is an improving ray for the dual RMP if and only if it satisfies the following conditions:

$$
\begin{aligned}
&-\sum_{m\in M}\Delta(\zeta)_m - \sum_{st\in D}\Delta(\eta)_{st} > 0, && && (8.45.0)\\
&\Delta(\gamma)_{kij} + \Delta(\gamma)_{jik} - \sum_{m\in M:\Lambda^i_{\{k,j\}}\subset m}\frac{\Delta(\zeta)_m}{C^i_{\{k,j\}}} = 0, && \forall \Lambda^i_{\{k,j\}}\subset E, && (8.45.1)\\
&\mathrm{FC}^{st}(p) \geq 0, && \forall st\in D, \forall p\in\sigma^{st}, && (8.45.2)\\
&\Delta(\zeta)_m \in \mathbb{R}_+, && \forall m\in M, && (8.45.3)\\
&\Delta(\gamma)_{kij}, \Delta(\gamma)_{jik} \in \mathbb{R}_+, && \forall \Lambda^i_{\{k,j\}}\subset E, && (8.45.4)\\
&\Delta(\eta)_{st} \in \mathbb{R}, && \forall st\in D. && (8.45.5)
\end{aligned}
\qquad . \qquad (8.45)
$$

where the Farkas coefficient is defined as:

$$
\mathrm{FC}^{st}(p) \coloneqq \sum_{(k,i),(i,j)\in p} -d^{st}\Delta(\gamma)_{kij} + \sum_{m\in M}\sum_{(i,j)\in p\cap m} d^{st}\frac{\Delta(\zeta)_m}{C_{ij}} + \Delta(\eta)_{st}. \qquad (8.46)
$$

If$(\Delta(\eta), \Delta(\gamma), \Delta(\zeta))$ is an improving ray for the dual RMP, it might violate the constraints of (8.45.2) for some $st \in D$ and $p \in \varphi^{st} \setminus \sigma^{st}$.

The pricing algorithm finds, for each demand $st \in D$, a path with the minimum Farkas coefficient. Such paths with negative Farkas coefficient are added to σ^{st}. Adding the corresponding cut repairs unboundedness of the dual RMP, and equivalently, adding the corresponding column in the RMP repairs its infeasibility.

If the Farkas pricing does not find any path, then $(\Delta(\eta), \Delta(\gamma), \Delta(\zeta))$ certifies that the dual master problem is unbounded, and we can conclude that the primal master problem is infeasible.

For a solution with a non-zero cyclic path, the left-hand side of the clique capacity constraint is always at most the value of another solution after removing the cycles. Therefore, the simple path improves feasibility. Therefore, the inverse of an optimal path in the extended graph is also simple in the original graph.

In summary, we have the following corollary for the correctness of the pricing algorithm.

Corollary 8.10. *The pricing algorithm finds a simple path to improve the RMP solution in polynomial time.*

Since the pricing problem is solved for several iterations at each node of the search tree, the above corollary shows the efficiency of our approach.

8.5 Branching rule

Branch-and-price (B&P) is a generalization of linear programming (LP) based branch-and-bound specifically designed to handle integer programming (IP) formulations that contain a huge number of variables. B&P applies column generation at every node of the branch-and-bound tree. Branching occurs when no columns with negative reduced costs are found, but the LP solution does not satisfy the integrality conditions.

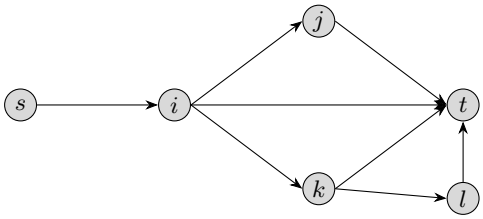


Figure 8.5 Paths diverge at i and k

The wUMCFC-P formulation contains binary path variables. Hence, at every node of the B&P tree, when the column generation converges to a fractional LP solution, the branching rule enforces the integrality of the path variables. Two branching rules can be applied: branching on edges or branching on paths.

Branching on paths is an easy way to set binary path variables to zero or one. If a variable is fixed to zero, the pricing algorithm might regenerate it. In the worst case, the solver generates a column, fixes it to zero, regenerates it again, and hence never terminates.

We choose the branching rule on edges. Branching on edges [36] forbids flows to use certain edges. For $st \in D$, an edge $e \in E$ is $st-$forbidden at a sub-B&P tree if e is forbidden to transmit any $st-$flow in any nodes of the sub-search tree. To forbid an edge e, we set to zero the existing columns of the paths containing e at the branching node, and forbid generation of any $st-$path containing e in its sub-search tree. Every node of the B&P tree records a set F^{st} of forbidden edges for each $st-$flow.

To be compatible with the proposed pricing algorithm, we can simply delete edges in F^{st} from the original G, and generate the extended graph EG from this subgraph of G. In this way, none of the $st-$paths through a forbidden edge would be generated.

Note that all the properties of the pricing algorithm still hold for the subgraph of G, and all the paths generated are simple in the subgraph and in G as well.

Let $\chi^{st} = \{p \,|\, y_p > 0,\, p \in \varphi^{st}\}$ be the set of $st-$path variables with non-zero values in the LP relaxation. Let T be the sub-graph of G supporting χ^{st}. In T, the in-degree of s is zero, the out-degree of t is zero, and the in-degree and the out-degree of the remaining nodes in T are identical.

For example, in Figure (8.5) there are 4 $st-$paths, 2 divergence nodes i and k, and i is the first divergence node.

In our implementation (8.1), we choose the flow $\overline{st} = \operatorname{argmax}_{st \in D:|\chi^{st}|>1} d^{st}$. Thus the branching rule has a strong impact on the dual bound. Note that branching on edges incident to the first divergence node fixes more active paths than branching on other edges.

Let the node i be the first divergence node of the tree T associated with $\overline{st}$.

Denote by $N_i^{\overline{st}} = \{(i,j) \,|\, (i,j) \in T\}$ the set of out edges at node i in T, by $O_i^{\overline{st}} = \{(i,j) \,|\, (i,j) \in E\} \setminus (F^{st} \cup N_i^{\overline{st}})$ the set of non-forbidden out edges of $E \setminus T$at node i. Note that $N_i^{\overline{st}} \cap O_i^{\overline{st}} = \emptyset$. Since the $st-$flow should be binary, it must take at most one edge of $N_i^{\overline{st}} \cup O_i^{\overline{st}}$ in its unique path; this forms a binary disjunction branch.

We partition $N_i^{\overline{st}}$ and $O_i^{\overline{st}}$ equally into $F_1^{\overline{st}}$ and $F_2^{\overline{st}}$, which are $\overline{st}-$forbidden edge subsets for two child nodes in the branch.

First, $N_i^{\overline{st}}$ is divided into two parts according to $\overline{st}-$flow values on its edges, such that the current $\overline{st}-$flow is partitioned into the two child nodes in a balanced way. Then, the partition of $O_i^{\overline{st}}$ is randomly chosen by order.

Algorithm 8.1: Branching rule

Data:
G: the original graph representing the network, $G = (V, E)$
χ: the set of path variables with non-zero values for each demand.
F: the set of forbidden edge sets at the current B&P node.
Result: Updated forbidden edge sets for two child nodes

$\overline{st} \longleftarrow \text{argmax}_{st \in D:|\chi^{st}|>1} d^{st}$
Construct the sub-graph T supporting $\chi^{\overline{st}}$
Find the first divergence node i of T
$N_i^{\overline{st}} \longleftarrow \{(i,j) \,|\, (i,j) \in T\}$
$O_i^{\overline{st}} \longleftarrow \{(i,j) \,|\, (i,j) \in E\} \setminus (F^{\overline{st}} \cup N_i^{\overline{st}})$
Compute values of $\overline{st}-$flow on the edges of $N_i^{\overline{st}}$ according to $\chi^{\overline{st}}$
Sort $N_i^{\overline{st}}$ in decreasing order by flow values on the edges
$F_1^{\overline{st}}, F_2^{\overline{st}} \longleftarrow \emptyset$
for $(i,j) \in N_i^{\overline{st}}$ **do**
    if $|F_1^{\overline{st}}| < |F_2^{\overline{st}}|$ **then**
        $F_1^{\overline{st}} \longleftarrow F_1^{\overline{st}} \cup \{(i,j)\}$
    else
        $F_2^{\overline{st}} \longleftarrow F_2^{\overline{st}} \cup \{(i,j)\}$
for $(i,j) \in O_i^{\overline{st}}$ **do**
    if $|F_1^{\overline{st}}| < |F_2^{\overline{st}}|$ **then**
        $F_1^{\overline{st}} \longleftarrow F_1^{\overline{st}} \cup \{(i,j)\}$
    else
        $F_2^{\overline{st}} \longleftarrow F_2^{\overline{st}} \cup \{(i,j)\}$
return $F_1^{\overline{st}}, F_2^{\overline{st}}$

8.6 Computational Results

In this section, we describe the test instances and evaluate the performance of the proposed algorithms. The path-based formulation wUMCFC-P of each instance is solved by the B&P algorithm. The compact wUMCFC-EB formulation is solved by the MILP CPLEX solver. The first experiments compare the performance of the B&P algorithm implemented with SCIP to the CPLEX solver performance.

The second experiment solves the wUMCF problems and the wUMCFC problems by the B&P algorithm to observe the impact of network coding. The wUMCF problem is obtained by setting $u = 0$ and deleting coding opportunity constraints from the wUMCFC problem.

8.6.1 Instances

We generate 40 instances and divide them into two classes, low-demand, and high-demand test beds. Each instance describes the graph G representing a WSN, the costs β of its edges, the demands D, the capacities C of its edges and the clique set M.

For each test bed, there are 10 subclasses, and each subclass contains 2 instances with the same number of nodes and demands. The number of nodes $|V|$ ranges from 30 to

120 by a step of 10, and each test bed contains instances of these 10 distinct sizes. The number of demands, $|D|$, equals $0.4|V|$ and $0.8|V|$ for low-demand and high-demand test beds, respectively.

Given the number of nodes and demands, the generation procedure follows as:

1. Generate a random bi-directed geometric graph $G = (V, E)$ of a given number of nodes in the unit square, where the Euclidean radius for linking two nodes is proportional to $\sqrt{\frac{1}{n}}$.
2. For each $i \in V$, sample the node cost β_i from the truncated standard normal distribution s.t. $0.8 \leq \beta_i \leq 1.2$.
3. For each $(i, j) \in E$, sample the capacity C_{ij} uniformly from $\{1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6\}$, and set the cost $\beta_{ij} = \frac{\beta_i}{C_{ij}}$.
4. For each demand $st \in D$: randomly select a pair of source and target nodes s and t, and sample the demand d^{st} from a normal distribution with a mean proportional to $\frac{1}{|D|}$.
5. Construct the conflict graph G_c of G according to the 2-dist interference model in Section 8.2.2.
6. Find the set M of maximal cliques in G_c using *Networkx's* recursive backtracking algorithm [169].

Note that finding all maximal cliques of a graph is an $\mathcal{NP}-$ complete problem. But the generated geometric graphs are sparse, the number of maximal cliques is not very large and the enumeration by *Networkx's* recursive backtracking algorithm is efficient. Therefore, we generate all the clique capacity constraints according to the 2-dist interference model, by using the clique set M computed by the *Networkx* algorithm.

For each coding opportunity set $\Lambda^i_{\{k,j\}} \subset E$, the parameters $\tau^i_{\{k,j\}}$ (for energy saving) and $C^i_{\{k,j\}}$ (for capacity increasing) are computed based on Section 8.2.3.

We only generate feasible and non-trivial instances. An instance is feasible if it has a solution and an instance is trivial if its LP relaxation root node is already feasible, i.e., satisfies the integrality constraints.

For each input, we repeat the above procedure until the instance is feasible and non-trivial.

The generated test instances are described in the first column of Table (8.3) and the first column of Table (8.4). The first letter indicates whether it belongs to the low or high-demand test bed (L or H), and the following **V.E.D** indicates respectively the numbers of nodes, edges and demands.

The B&P algorithm in our solver *wUMCFC* is implemented using SCIP (version 7.0.2) [147] with CPLEX (version 12.10.0) as an LP solver. We use CPLEX in the single-thread mode (without parallelism) to solve the compact edge balance formulation. The open source solver *wUMCFC* and test instances can be found in our project web page: github.com/lidingxu/wUMCFC.

The computing environment has an Intel Core i7-6700K CPU at 4.00 GHz and 16 GB of RAM under Ubuntu 20.04 system. The time limit is set to 3600 CPU seconds. The relative duality gap tolerance is set to 1e-4.

8.6.2 Numerical comparisons of formulations

In this section, we perform the numerical comparison between the path-based formulation wUMCFC-P and the compact edge balance formulation wUMCFC-EB. We evaluate performance metrics of the B&P algorithm for the wUMCFC-P formulation and CPLEX for the wUMCFC-EB formulation.

We implement the pricing algorithm and the branching rules for the B&P algorithm in SCIP; the initial columns are the shortest paths of the demands.

Table (8.3) and Table (8.4) report the results for low and high-demand data sets respectively.

For the B&P algorithm and CPLEX, we report the primal bound $\overline{z^*}$, the relative duality gap in percentage, the run time t in CPU seconds, the number of variables, the number of constraints, and the number of nodes. Additionally for the B&P algorithm, we also record the pricing time t_P in CPU seconds, the number of pricing calls and the number of generated paths.

Among all 40 instances, the B&P algorithm solves 13 instances to optimality, and finds primal feasible solutions for all 40 instances. CPLEX solves 10 instances to optimality and finds primal feasible solutions for 23 instances. This result can be explained by the fact that the size of the edge-based formulation is significantly larger than the size of the path-based formulation, especially for large networks and high-demand instances. In the edge-based formulation, the number of constraints grows linearly with the number of demands and the size of the network, while the number of constraints of the path-based formulation mainly depends on the size of the network. Both the edge-based formulation and the path-based formulation need auxiliary continuous variables to model the coding opportunities, but the edge-based formulation has additional variables q on incident edges. The number of variables of the edge-based formulation is $O(|E|^2|D|)$, and the number of variables of the path-based formulation is $O(|E|^2)$ in addition to the path variables.

Even with the generated columns (path variables), the size of the path-based formulation is still smaller than the size of the edge-based formulation. This occurs because, for each demand, only a small part of the path variables are non-zero in the optimal solution. Columns generated, at one node in the search tree, are not necessarily included in the LP relaxation of a different node. If some columns are equal to zero in the LP relaxation, SCIP can remove these columns from descendant nodes, and add them back dynamically. Hence, the size of the LP is kept as small as possible.

To compare the performance of the B&P algorithm and CPLEX, we compute the shifted geometric means (by 10s) of run times for low-demand, high-demand and all instances respectively. The run time of unsolved instances is taken as the time limit (3600 seconds). For the B&P algorithm, the mean times for low-demand, high-demand and all instances are respectively: 588.3 seconds, 1365.2 seconds and 897.1 seconds. For CPLEX, the mean run times for low-demand, high-demand and all instances are respectively: 1005.9 seconds, 1798.3 seconds and 1345.4 seconds. Therefore, the B&P algorithm is respectively 0.71, 0.31 and 0.5 times faster than CPLEX on low-demand, high-demand and all instances.

We also compare the average duality gaps for the instances in which feasible solutions are found by CPLEX. The average duality gap for B&P is 0.62% and the average duality gap for CPLEX is 1.04%.

In summary, B&P outperforms CPLEX in run time and the quality of solutions.

The two following tables show that a significant part of the total running time is dedicated to solving the pricing problem. Let $\frac{t_p}{t}$ be the ratio of the pricing time over total run time. On average, the ratio is 26%, and it increases with the size of the network and the number of demands.

Table 8.3 Solver performance on low-demand test bed

Instance	B&P for the wUMCFC-P									CPLEX for the wUMCFC-EB					
	$\overline{z^*}$	Gap(%)	t	t_p	#calls	#paths	#vars	#cons	#nodes	$\overline{z^*}$	Gap(%)	t	#vars	#cons	#nodes
L30.152.12	2.72	0.0	1.1	0.1	499	1156	1532	755	95	2.72	0.01	8.08	10909	4562	41
L30.166.12	2.18	0.0	4.8	1.1	1242	2673	3139	940	226	2.18	0.01	2.28	13444	5078	5
L40.240.16	2.26	0.0	3.5	0.5	486	1839	2563	1487	29	2.26	0.01	16.27	28197	9719	0
L40.260.16	2.37	0.0	43.5	9.6	1724	6307	7234	1924	150	2.37	0.01	156.58	34576	10739	46
L50.308.20	3.06	0.01	25.9	6.2	1766	5422	6391	1975	131	3.06	0.0	246.16	47400	15461	40
L50.334.20	2.67	0.0	30.9	7.7	1480	5802	7071	2595	64	2.67	0.01	76.34	59215	16857	6
L60.394.24	4.39	4.85	3600.1	1931.0	23524	70034	71426	2841	1735	4.59	9.92	3600.03	80568	23468	89
L60.342.24	3.87	1.0	3600.2	2296.7	33058	55133	56101	2015	7586	3.84	0.27	3600.02	58671	20109	1119
L70.446.28	4.09	0.57	3600.0	2214.9	30522	74835	76320	3002	2648	4.09	0.57	3600.03	102021	30482	298
L70.380.28	3.96	0.01	478.6	195.9	9299	24732	25769	2136	1610	-	-	3600.02	74036	26123	155
L80.482.32	3.77	0.01	72.8	19.0	2587	8362	9903	3145	187	3.78	0.32	3600.08	123478	37761	90
L80.672.32	2.51	0.52	3600.2	1127.5	5111	31152	34772	7781	131	2.57	3.59	3600.2	263385	54060	0
L90.544.36	6.44	1.83	3600.1	1311.4	25327	43417	45040	3376	5685	6.4	0.58	3600.07	149607	47719	8
L90.552.36	5.97	0.69	3600.1	1682.6	27540	60768	62509	3549	3476	5.97	0.82	3600.05	156861	48240	10
L100.716.40	4.16	1.68	3600.2	1171.6	15791	68773	71573	5681	1065	-	-	3600.08	270651	68860	0
L100.568.40	6.18	1.46	3600.2	1332.3	19217	52610	54258	3463	3327	6.21	1.57	3600.15	167922	54855	0
L110.710.44	5.98	2.02	3600.0	714.6	24847	39220	41461	4623	10531	-	-	3600.18	248479	74770	0
L110.710.44	6.11	1.6	3600.0	942.9	17980	60739	63001	4625	2486	6.26	4.07	3600.12	251989	74860	0
L120.708.48	4.95	0.34	3600.0	946.6	41914	45198	47283	4221	19147	-	-	3600.07	256114	81248	35
L120.698.48	6.28	0.45	3600.1	1144.6	35174	49818	51782	3994	12351	6.27	0.22	3600.08	245289	80517	3

8.6.3 Effects of network coding

According to Section 8.2.3, network coding has two effects: capacity increasing and energy saving.

The first effect enables more feasible routes. It depends on networks' bottlenecks, capacities and demands. Thus, network coding is more useful for instances where demands are large and capacities are limited.

The second effect yields fewer transmissions. Recall that the saved energy equals the CSC z_c. This effect reduces the energy cost of every feasible route, which depends on the costs of edges.

The wUMCFC problem is an UMCF problem integrating clique capacity constraints and network coding, and the wUMCF problem is the UMCF problem only with clique capacity constraints.

Since in the previous experiments, the B&P algorithm for the path-based formulation outperforms CPLEX for the edge balance formulation, we compare the objective values (energy costs) of the wUMCF and the wUMCFC problems by solving their path-based formulations.

The path-based formulation of the wUMCF problem, i.e., wUMCF-P, is obtained by removing coding variables and coding constraints from wUMCFC-P. In our starting experiment, the following instances have solutions for the wUMCFC (with coding) formulation but have no solutions for the wUMCF (without coding) formulation:

- low-demand: L30.166.12, L50.308.20, L110.710.44, L120.708.48 and L120.698.48.
- high-demand: H30.132.24, H40.226.32, H50.224.40, H50.282.40, H90.552.72, H90.504.72 and H110.630.88.

Table 8.4 Solver performance on high-demand test bed

Instance	B&P for the wUMCFC-P									CPLEX for the wUMCFC-EB					
	$\overline{z^*}$	Gap(%)	t	t_p	#calls	#paths	#vars	#cons	#nodes	$\overline{z^*}$	Gap(%)	t	#vars	#cons	#nodes
H30.146.24	2.56	0.0	0.8	0.1	301	880	1222	688	56	2.56	0.01	29.78	20150	8371	16
H30.132.24	3.12	0.01	0.3	0.1	195	580	859	560	45	3.12	0.01	8.22	16700	7564	7
H40.220.32	3.56	0.01	123.6	50.2	5330	12931	13545	1261	962	3.56	0.01	599.12	47152	16505	193
H40.226.32	2.69	0.01	195.3	42.7	8675	14541	15195	1306	3453	2.69	0.01	370.43	49522	16680	284
H50.224.40	4.81	0.13	3600.0	1040.3	126298	25862	26444	1165	110427	4.81	0.21	3600.01	58325	23213	2302
H50.282.40	4.15	0.0	821.7	316.9	15530	37497	38313	1642	3607	-	-	3600.02	78667	26351	146
H60.394.48	4.61	0.69	3600.0	1708.4	27550	52328	53725	2815	6622	-	-	3600.04	158063	43869	23
H60.362.48	3.45	1.25	3600.0	1574.0	16715	52420	53635	2525	2411	-	-	3600.05	137815	40112	3
H70.452.56	4.91	7.53	3600.0	839.7	7596	54355	56049	3543	432	-	-	3600.08	223788	58485	0
H70.464.56	4.02	0.47	3600.0	1593.2	15744	55929	57557	3354	2450	4.02	0.69	3600.15	217070	60690	0
H80.552.64	4.42	2.11	3600.1	931.8	19115	56330	58030	3458	4560	-	-	3600.16	264689	77730	0
H80.484.64	4.54	5.04	3600.0	692.5	21358	39771	41294	3127	7455	-	-	3600.12	236580	71154	0
H90.552.72	5.52	0.86	3600.0	710.7	26380	40918	42670	3523	12196	-	-	3600.12	308703	92601	0
H90.504.72	4.91	0.74	3600.0	777.9	31117	39608	41038	2894	15619	4.92	1.1	3600.12	257387	84783	7
H100.594.80	4.71	1.26	3600.1	1090.4	22752	55539	57394	3747	6329	-	-	3600.1	366192	111618	0
H100.598.80	4.59	1.67	3600.2	1069.0	18056	47139	48945	3741	4337	-	-	3600.14	359387	112133	0
H110.772.88	5.61	3.51	3600.0	815.8	8290	53669	56688	6143	470	-	-	3600.26	630648	156050	0
H110.630.88	7.01	2.57	3600.0	885.1	22104	38323	40105	3619	5209	-	-	3600.13	397273	130245	0
H120.720.96	7.18	2.13	3600.0	586.0	21033	40337	42621	4601	7588	-	-	3600.21	542513	161613	0
H120.740.96	6.25	1.56	3600.0	429.2	13977	38935	41304	4821	4206	-	-	3600.48	562586	165990	0

Demands of these instances exceed the capacity of the networks, but network coding could increase the capacity. As a result, more feasible routes are possible, and these instances have solutions under the wUMCFC formulation. These instances demonstrate the first effect of network coding, i.e., increasing capacities, which improves QoS of WSNs.

We evaluate the energy cost only on feasible instances. A binary search method is used to find feasible demands for each of these infeasible instances. Revised instances have a suffix 'R' appended to their labels, and we replace original instances with revised instances in new test beds. We still set the time limit to 3600 seconds. B&P might produce primal solutions with unclosed duality gaps at the end of the time limit. The subsequent experiment shows that network coding reduces the energy cost significantly although the unclosed duality gaps yield estimation errors.

Recall that the objective function of the wUMCF problem (denoted by z) is the first term of the objective function of the wUMCFC problem (denoted by $z\_$), and their difference $z_c = \sum_{\Lambda^i_{\{k,j\}} \subset E} \tau^i_{\{k,j\}} u^i_{\{k,j\}}$ is the energy cost explicitly saved by network coding (CSC). For each instance, we report the incumbent value $\overline{z^*}$ of the wUMCF problem (resp. $\overline{z^*_-}$ of the wUMCFC problem), the relative duality gap in percentage, the run time t in CPU seconds, the number of variables and the number of constraints. For the wUMCFC problem, we also record CSC $\overline{z^*_c}$ of the incumbent solution. The results are reported in Tables (8.5) and (8.6) for low and high-demand test beds respectively.

Let z^* and z^*_- be the optimal values of the wUMCF and the wUMCFC problems respectively, B&P could compute them with unlimited time and memory. Define by $f = \frac{z^* - z^*_-}{z^*} = 1 - \frac{z^*_-}{z^*}$ the relative energy saved by network coding, and $\hat{f} = 1 - \frac{\overline{z^*_-}}{\overline{z^*}}$ is the estimator of f.

Given the relative duality gap and the primal bound (incumbent value), we can compute the dual bound of the optimum.

Let $\underline{z^*}$ be the lower bound on the wUMCF problem so the error of the estimator follows

$$f - \hat{f} = \frac{z^* \, \overline{z^*_-} - \overline{z^*} \, z^*_-}{z^* \, \overline{z^*}} \geq \frac{\underline{z^*} \, \overline{z^*_-} - \overline{z^*} \, \overline{z^*_-}}{\overline{z^*} \, \underline{z^*}} := \hat{e}, \tag{8.47}$$

for which $\hat{e}$ gives a lower bound on the error.

The average relative energy saving is 18.35%, and the average error lower bound is -0.04%. Therefore, the non-closed duality gap just induces a negligible estimation error of the relative energy saving. The network coding saves energy cost significantly.

Increasing capacity could yield a larger feasible route set, hence the optimum decreases. Indeed, the CSC contributes to the main part of the energy saving. For each instance, we compute the first term of the objective function of the wUMCFC problem, i.e., $\widetilde{z^*_-} = \overline{z^*_-} + \overline{z^*_c}$, which is the energy cost of the given incumbent routing without network coding. Among 62.5% instances, $\widetilde{z^*_-} > \overline{z^*}$, only the second effect (CSC) can contribute to energy saving. Among the remaining 37.5% instances, $\overline{z^*} \geq \widetilde{z^*_-} > 0.973\overline{z^*}$; the energy cost without saving ($\widetilde{z^*_-}$) is close to the energy cost of the wUMCFC problem ($\overline{z^*}$), therefore the first effect (capacity increasing) still has a minor contribution to energy saving. The ratio $\frac{\overline{z^*_c}}{\widetilde{z^*_-}}$ is CSC over unreduced energy cost, its average value among all instances is 19.16%. Consequently, we can conclude that CSC has a major contribution to energy saving.

However, with network coding the number of variables and constraints of the wUMCFC problem are significantly larger than those of the wUMCF problem for the same graph. Hence, the solution times and the duality gaps of the wUMCFC problems increase with the size of instances. To achieve better performance, the wUMCFC problem requires a faster algorithm. An adapted branch-and-price algorithm is developed in this chapter.

Table 8.5 Effects of network coding on low-demand test bed

Instance	wUMCF					wUMCFC					
	$\overline{z^*}$	t	Gap (%)	#vars	#cons	$\overline{z^*_-}$	$\overline{z^*_c}$	t	Gap (%)	#vars	#cons
L30.152.12	3.0	0.0	0.0	33	27	2.72	0.37	1.1	0.0	1487	755
L30.166.12.R	2.42	0.0	0.01	32	32	1.76	0.34	0.1	0.0	620	940
L40.240.16	2.37	0.0	0.0	55	71	2.26	0.2	3.3	0.0	2445	1487
L40.260.16	2.64	0.2	0.0	175	102	2.37	0.36	42.9	0.01	7101	1924
L50.308.20.R	2.86	0.2	0.0	155	77	2.47	0.3	39.5	0.0	7214	1975
L50.334.20	3.0	0.1	0.0	77	97	2.67	0.41	40.1	0.0	7818	2595
L60.394.24	5.33	51.1	0.01	1856	105	4.39	1.35	3600.0	4.92	66727	2841
L60.342.24	4.69	0.9	0.01	347	127	3.87	0.77	3600.2	1.02	52664	2015
L70.446.28	4.59	0.2	0.01	121	88	4.09	0.6	3600.0	0.58	71244	3002
L70.380.28	4.62	2.9	0.01	481	118	3.96	0.77	502.7	0.01	25769	2136
L80.482.32	4.08	10.2	0.01	869	127	3.77	0.3	74.7	0.01	9903	3145
L80.672.32	2.83	0.0	0.0	32	605	2.51	0.54	3600.5	0.52	34712	7781
L90.544.36	7.57	2.4	0.01	282	202	6.44	1.6	3600.0	1.89	36228	3376
L90.552.36	7.77	3600.0	0.46	5773	139	5.97	1.9	3600.0	0.78	49755	3549
L100.716.40	5.07	1.2	0.0	229	161	4.17	0.98	3600.1	1.99	68123	5681
L100.568.40	7.02	0.8	0.0	186	247	6.18	1.12	3600.0	1.47	51570	3463
L110.710.44.R	8.15	3600.0	0.91	10055	229	5.51	1.98	3600.1	1.59	40143	4623
L110.710.44	7.56	88.9	0.01	718	189	6.11	1.73	3600.0	1.6	61856	4625
L120.708.48.R	5.18	0.6	0.01	220	147	4.13	1.11	3600.0	0.13	31142	4221
L120.698.48.R	6.47	3600.1	0.06	484	162	5.17	1.23	3600.0	0.32	39410	3994

8.7 Conclusion

In this article, we propose a mathematical programming approach to address the problem of optimizing the energy cost of data transmission in multi-hop WSNs. We propose new formulations of the problem that take into account specific technical constraints of wireless communication, such as unsplittable routing, interference and network coding.

Table 8.6 Effects of network coding on high-demand test bed

Instance	wUMCF					wUMCFC					
	$\overline{z^*}$	t	Gap (%)	#vars	#cons	$\overline{z_-^*}$	$\overline{z_c^*}$	t	Gap (%)	#vars	#cons
H30.146.24	3.36	0.0	0.0	97	52	2.56	0.79	0.7	0.0	1258	688
H30.132.24.R	2.84	0.6	0.0	223	50	2.51	0.32	0.3	0.01	745	560
H40.220.32	4.01	0.1	0.01	118	97	3.56	0.48	132.8	0.01	14000	1261
H40.226.32.R	2.45	0.2	0.01	101	62	2.1	0.34	34.8	0.01	6671	1306
H50.224.40.R	5.37	102.3	0.01	1332	81	4.51	0.92	1563.0	0.01	15351	1165
H50.282.40.R	5.44	3600.0	0.4	3866	90	3.97	1.01	233.7	0.01	19707	1642
H60.394.48	5.32	33.0	0.01	1511	117	4.61	0.7	3600.0	0.64	59957	2815
H60.362.48	4.04	2.0	0.01	442	191	3.45	0.85	3600.0	1.3	51053	2525
H70.452.56	5.94	3600.0	0.07	1979	267	4.91	1.35	3600.0	7.53	54468	3543
H70.464.56	4.66	0.5	0.01	192	210	4.02	0.71	3600.0	0.48	56404	3354
H80.552.64	5.72	0.6	0.01	271	186	4.42	1.54	3600.2	2.12	56716	3458
H80.484.64	6.24	3600.0	0.31	1933	209	4.54	1.63	3600.0	5.08	36467	3127
H90.552.72.R	6.07	3600.1	0.03	929	163	4.5	1.47	3600.0	0.38	27090	3523
H90.504.72.R	6.68	3600.1	0.17	674	178	4.66	1.94	3600.0	0.84	35466	2894
H100.594.80	5.93	1.1	0.01	235	197	4.71	1.29	3600.0	1.26	56155	3747
H100.598.80	6.09	3.5	0.01	539	289	4.59	1.64	3600.1	1.67	48364	3741
H110.772.88	7.4	2.0	0.01	289	281	5.61	1.78	3600.0	3.51	56031	6143
H110.630.88.R	7.47	1.3	0.01	329	231	5.33	2.36	3600.0	2.08	40768	3619
H120.720.96	9.95	996.2	0.01	1935	225	7.18	2.88	3600.0	2.13	42412	4601
H120.740.96	8.25	1.9	0.01	290	275	6.25	2.26	3600.0	1.56	41935	4821

A column generation approach is developed as well as a branch-and-price framework to solve this $\mathcal{NP}$-hard and challenging problem. The numerical experiments show that the proposed branch-and-price algorithm outperforms the CPLEX solver both in terms of running time and duality gap. We also show that, for all instances, network coding reduces energy cost significantly. For hard instances, it enables more feasible routes with lower objective function values. The computational efficiency of the branch-and-price algorithm is improved by adapted branching rules and solution tracking mechanisms.

In the future, we plan to integrate more complex coding schemes and optimize the number of activated devices in the network (see [297]). This integration may increase the complexity of the problem and introduce more binary decision variables in the models. We aim to develop a polyhedral study of this problem, derive new valid inequalities and develop branch-and-price-and-cut algorithms.

We also plan to extend this chapter, from general WSNs to the special cases of IoT deployments. Connecting IoT devices through WSNs requires specific technologies. These technologies induce more technical constraints on routing, resource management and network coding protocols. We are interested in extending our models to integrate the specific constraints of the IoT deployment on multi-hop WSNs.

The further study of such optimization problems and the development of corresponding decision-support tools will help to offer new services with increased quality of service in future wireless networks deployments.

Chapter 9

Piece-wise linear modelling in continuous covering on networks

9.1 Introduction

Covering in Operations Research refers to the optimization problem of deciding the location of facilities to “cover" the points of the so-called demand set, which should fall within the radius coverage of at least one of the installed facilities. This classic problem finds applications in many different domains, including health care [7], surveillance of transport networks [167], computer networks security [278], crane location for construction [73], military evacuation systems [183], homeland defense [45], and urban air mobility [311].

Covering problems have taken many forms in the literature. A rough classification distinguishes between *maximal covering* location and *set-covering* location problems. The former aims at maximizing the covered demand with a fixed number of facilities (see e.g. [92]), while the latter seeks to minimize the number of installed facilities to cover all the demand (see e.g. [288]). In these classic works [92, 288], the problem is defined on a network and both demand points and candidate facility locations are at nodes. Most of the variants of network covering studied afterward consider at least one of these two sets to be finite, see the reviews [254, 148] and the references therein. However, this assumption corresponds to ideal but usually unrealistic scenarios (the reader is referred to the real applications of the above paragraph). As an example, in the eVTOLs safety landing site location problem [311], a set of emergency landing sites has to be installed on the traffic network in such a way that any point of the same is covered. Something similar happens for the location of ambulance bases in rural areas studied in [7]. Some works addressing network covering with continuous sets of both candidate locations and demand points are [68, 56, 33], for maximal covering, and [166, 142, 174], for set-covering. We focus on the latter variant, which we call the continuous set-covering problem.

Gurevich et al. [166] presented an algorithm to compute an optimal continuous set-covering when the covering radius and the edge’s lengths are natural numbers. This algorithm is polynomial time for the class of networks satisfying that every non-separable component is either an edge, a simple cycle, or a simple cycle with one chord, that is, for “almost tree” networks. More recently, Fröhlich et al. [142] also studied the same version of the continuous set-covering with natural numbers. The authors presented three different approaches to solve

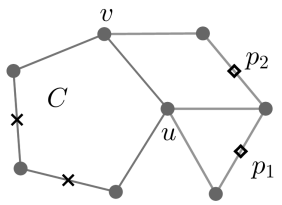


Figure 9.1 Two cycle coverage points with respect to a cycle C of five nodes

the problem, including a MILP formulation. On the other hand, Hartmann et al. [174] focused on the computational complexity of the continuous set-covering for general covering radii. They proved that, when all edges have unit length, the continuous set-covering is polynomially solvable if the covering radius is a unit fraction, and is NP-hard otherwise.

We can now formally state our problem. Consider an undirected connected network $N = (V, E, l)$, where $l : E \to \mathbb{R}_+$ is the edges' length function. We will denote $l_e := l(e)$ the length of e. The continuum of points on all edges and nodes of N is denoted with $C(N)$. The distance function $d(\cdot, \cdot)$ defines the distance between two points, which coincides with the length of the shortest path in $C(N)$ connecting them. Given $\delta > 0$, a point $p \in C(N)$ is said to δ-cover $p' \in C(N)$ (respectively, p' δ-covers p) if $d(p, p') \leq \delta$ holds. The parameter δ is called the covering radius. The continuous δ-covering location problem on N is to find a set of facility locations in $C(N)$ of minimum cardinality that δ-covers the whole network, and is formally stated next.

Definition 9.1 (Continuous Set-Covering Problem (CSCP))**.** *The Continuous Set-Covering Problem on a network N can be expressed as the following optimization problem:*

$$\min \Big\{ |\mathcal{P}| : \ \mathcal{P} = \{p_i\}_{p_i \in C(N)} \text{ and } \forall p \in C(N), \ \exists p_i \in \mathcal{P} \text{ s.t. } d(p, p_i) \leq \delta \Big\}. \tag{9.1}$$

A set $\mathcal{P}$ satisfying the condition within (9.1) *is called a δ-cover of N, while $\mathcal{P}^*$ minimizing* (9.1) *is a minimum δ-cover.*

The set $\mathcal{P}$ in Definition 9.1 can represent the locations of ambulance bases [7], surveillance cameras [167], routing servers in a network of computers [278], cranes for construction [73], aerial military medical evacuation facilities [183], aircraft alert sites for homeland defense [45], or eVTOL safety landing sites in an urban area [311].

The CSCP is known to be NP-hard, see [174]. Due to the continuous nature of CSCP, there is an infinite number of candidate locations. Previous works reduce CSCP to a tractable set covering problem by discretization. A first observation is that typical simplifications proposed in other related studies are not valid for the CSCP.

Discretization methods identify finite dominating sets (FDS), which are finite subsets of candidate locations guaranteed to contain an optimal solution. From the literature, we know at least three FDS for related variants of the CSCP, which rely on different assumptions on the network and the covering radius. Here, we review and compare these FDS. Since we aim to propose a general exact algorithm, we show that it may not be viable to extend discretization methods to solve the CSCP for general networks and real radii.

First, Church and Meadows [91] studied the problem with demand at nodes, and identified the following points:

$$NIP := \{p \in C(N) :\ d(p,v) = \delta \text{ for some } v \in V\}.$$

The authors proved that $FDS_1 := V \cup NIP$ is an FDS for the network set-covering problem when the set of demand points is V and that of candidate locations is $C(N)$.

Secondly, Gurevich et al. [166] studied the continuous set-covering problem when the covering radius and the edge's lengths are natural numbers. They presented an FDS for the case of all edge lengths being one, which can be easily extended to the case of general edge lengths (see [142]),

$$FDS_2 := \left\{p \in C(N) :\ d(p,v) = \frac{i}{2 \cdot l_e} \text{ for some } e \in E \text{ and } v \in e;\ i = 0, \ldots, 2 \cdot l_e\right\}.$$

Note that FDS_2 depends on the edge's length.

Lastly, Fröhlich et al. [142] proposed a different FDS for the same version of the continuous set-covering with natural numbers. The authors defined the following set of cycle coverage points:

$$CCP := \left\{p \in C(N) :\ d(p,C) := \min_{y \in C}\{d(p,y)\} = \left(\delta - \tfrac{l_C}{2}\right) \bmod \delta,\ p \notin C,\ \text{for a simple cycle } C \subseteq C(N)\right\},$$

where l_C is the total length of the cycle C. Suppose that a cycle C is covered by a set of facilities. A cycle coverage point is the furthest point where a facility that contributes to cover C can be moved without compromising the coverage of the cycle (if the rest of the facilities remain unchanged).

Fig. 9.1 illustrates this idea. In the depicted example, all edges have unit length and $\delta = 2$. The figure depicts p_1 and p_2, which are CPP with respect to the cycle C of five nodes. Note that $d(p_1, C) = d(p_2, C) = 1.5 (= (2 - 5/2) \bmod 2)$. Fig. 9.1 also depicts two locations in C (marked with symbols 'x'), which correspond to two possible feasible locations for the remaining facility needed to cover C (note that the one at the bottom only yields a covering of the cycle if p_2 is located, while the other one together with either p_1 or p_2 can completely cover C).

The authors of [142] gave the following recursive definition of an FDS for the problem with natural numbers:

$$S_1 := V \cup NIP \cup CCP;$$

$$S_{j+1} := S_j \cup \{p \in C(N) :\ d(p,y) = \delta \text{ for some } y \in bd(\mathcal{A}(S_j))\};$$

$$FDS_3 := S_{|J|},$$

where $bd(\mathcal{A}(S_j))$ is the boundary of the area covered by S_j, and $J \subseteq E$ is the subset of edges to be covered. As the author explained themselves, $FDS_3 \subseteq FDS_2$. However, the cardinality of FDS_3 may be exponential in the input size, as the number of cycles in a network is in general exponential.

The example depicted in Fig. 9.2 illustrates that none of FDS_1 and FDS_2 are FDS for the CSCP. A similar observation was already presented in [56] for a related problem. The figure shows eight nodes on a path, where all edges have equal lengths. If $l_e = 1$ for all $e \in E$ and $\delta = 1.2$, $\mathcal{P} := \{p_1, p_2, p_3\}$ is an optimal δ-cover.

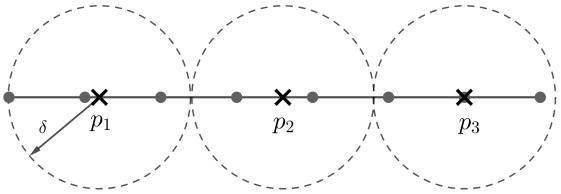


Figure 9.2 An instance of CSCP such that not all facilities in $\mathcal{P}^*$ are at a distance δ from some node

It can be easily observed that there is no optimal solution in which p_2 is placed either at a node or at a distance δ from some of the eight nodes, which shows that FDS_1 is not a valid FDS. On the other hand, it is also easy to check that there is not a feasible solution with the three facilities located either at nodes or middle-points of edges, which proves that FDS_2 is also not a valid FDS. As opposed to FDS_2, the assumption of the edge lengths and coverage radius being natural numbers is not fundamental in the definition of FDS_3. Conversely, FDS_3 is based on the idea of identifying those points at the "boundaries" of coverage areas, i.e., those delimiting the transition from covering/not covering a specific part of the network.

Consequently, FDS_3 could be extended to the general CSCP. However, such an extension potentially yields sets with many more candidates, due to the recursive construction of FDS_3 based on the distance function. Note that, if δ and the edge lengths are natural numbers, FDS_3 only contains points of the set

$$INT := \{p \in C(N) : d(p, v) \text{ is integer or half-integer for some } v \in V\}.$$

Indeed, if $\delta \in \mathbb{N}$, $NIP \subseteq INT$ is clear; $CCP \subseteq INT$ holds since the operation $(\delta - l_C/2)$ mod δ only yields half integers; and $FDS_3 \subseteq INT$ then easily follows by definition. However, if $\delta \in \mathbb{R}$, the locations of the points in FDS_3 are a priori undetermined, and its cardinality increases. Take the same example depicted by Fig. 9.1. If $\delta = 2.1$ (i.e. we increase δ just by 0.1), the CCP with respect to the cycle C of the Figure increases from two to four points.

9.1.1 Literature review

Facility location and set covering problems have many variants and applications in operations research and management science. Related literature to this work is vast; here we review a selection of works related to CSCP. In [265], the model allows that an edge is covered jointly by two facilities. In [269], the authors presented a unified vision of the common characteristics of facility location problems in a continuous space. In [257], the authors summarized the research progress in facility location problems on networks. A recent survey [302] provided a comprehensive overview of emergency facility location problems in logistics, including mathematical models, applications, and the commonly used solution methods. One of the most distinguishing features of variants of the maximal covering location problem is the solution space: continuous [254, 34], discrete [148, 105], or on networks [74, 57]. Especially in [57], the authors introduced the maximal covering location problem with edge demand. In [32], the authors studied the upgrading version of the maximal covering location problem with edge length modifications on networks. A related problem that has been recently studied is the obnoxious facility location problem [128]. It aims at locating undesirable facilities that have

a negative impact on communities. The most common objective is to maximize the shortest distance to the closest facility, and the problem has various variants featuring multiple facilities on the plane [129, 186], p-median objective [187], or edge demand on networks [57]. We refer to [93] for a recent review on the obnoxious facility location problem. For more related works, we refer to [9, 246, 144, 55, 179].

9.1.2 Contribution

As opposed to discretization methods, we directly tackle the CSCP for general networks and real radii. Our main contribution is an exact integer programming approach for the CSCP, together with tailored algorithms and strategies to tackle it. Even if this problem has been known for decades, surprisingly, only a few partial results are known for some special cases and sub-classes of networks. To the best of our knowledge, only one MILP model [142] has been proposed so far which can address the general CSCP. Such a model can be applied to any network whose edges do not measure more than the covering radius. This condition does not restrict the applicability of the MILP in [142], as any network can be transformed into an equivalent one that satisfies it.

Here, we present an enhanced MILP formulation that relies on the same assumption as that in [142], but whose numbers of constraints and variables have smaller order of magnitude. In addition, preprocessing strategies to reduce the number of variables of the model are studied, and tailored algorithms are presented. Approaches to strengthen this formulation are also presented, including big-M constants tightening and valid inequalities. The valid inequalities are constraints that reduce the feasible space without removing model solutions.

The introduction of a second MILP, which is scalable concerning the edge's lengths completes the main contributions of this work. This second MILP is an adaptation of the first one we propose, with the difference that it does not require all edge lengths to be smaller than the covering radius. Finally, our computational experiments prove that the MILP model in [142] is not scalable. On the other hand, the preprocessing technique drastically reduces the size of the first model proposed herein.

Finally, we show in the experiments that the second model we propose is superior to both the model from [142] and our first model, in terms of the solution quality and solving time.

In the proposed setting, both the candidate facility locations and the demand points are continuous sets (in particular, they coincide with $C(N)$). The problem could be defined for a subset of demand edges, $J \subseteq E$, and/or a subset of candidate locations $H \subseteq E$. The theoretical results, model, and methods described in this work apply to such cases, after straightforward adaptation.

9.1.3 Outline the chapter

The rest of the chapter is organized as follows. Sect. 9.2 presents useful notation and the theoretical development upon which our model is built. Then, our first MILP model is introduced in Sect. 9.3, while strategies to strengthen this model are described in the next section. The network processing algorithms that complement our MILP are detailed in Sect. 9.5. A second MILP model, which we call reduced formulation and is a modification of the first MILP, is presented in Sect. 9.6. Finally, Sect. 9.7 describes our computational experiments, and reports and analyzes the obtained results. Sect. 9.8 closes the chapter with some conclusions.

9.2 Covering characterization

This section presents several notation, definitions, assumptions, observations, and results related to the CSCP. On the one hand, Prop. 9.5 gives a characterization of the δ-covers of a network, which is based on the individual coverage of each edge of the network. Then, this result is refined to obtain a second necessary and sufficient covering condition in Prop. 9.9. It distinguishes between two alternative possibilities for covering each edge, namely complete or partial, and will be useful for our MILP formulation and methods. The rest of the section is oriented to characterize the so-called partial and complete covers. The idea of these sets is to delimit the areas of the network where a facility, if placed, would completely cover a given edge, and those where a facility would reach the edge (but maybe not completely cover it).

We first introduce some related notation, definitions and assumptions. We assume that V is totally ordered by the binary relation $\preceq$. Every edge $e \in E$ has a unique representation, $e = (v_a, v_b)$, where $v_a, v_b \in V$, and $v_a \preceq v_b$. From now on, we take $e = (v_a, v_b)$ indifferently as a continuum in $C(N)$ or as an edge ending at v_a, v_b. We extend the edges' length function to $l : C(N) \to \mathbb{R}_+$ as a length measure on the continuum of points. For two points $p, p' \in C(N)$, we denote by $\Pi(p, p') \subseteq 2^{C(N)}$ and $\Pi^*(p, p') \subseteq \Pi(p, p')$ the set of paths and shortest paths, respectively, connecting p and p'. Any path $\pi \in \Pi(p, p')$ is indifferently treated as a continuum in $C(N)$, then $l_\pi := l(\pi)$ is the length of π. The distance between p and p', $d(p, p')$, is the length of a shortest path connecting them:

$$d(p, p') := \min\{l_\pi : \pi \in \Pi(p, p')\} = l_{\pi^*} \text{ for any } \pi^* \in \Pi^*(p, p').$$

In particular, if p and p' belong to the same edge, we denote by $l(p, p')$ the length of the unique path in that edge connecting them. We work under the following assumption:

Assumption 9.2. *$\delta \geq l_e$ for all $e \in E$.*

If Assumption 9.2 did not hold, we could consider a set $I \subset 2^{C(N)}$, that would contain, for each $e = (v_a, v_b) \in E$, the following continuum sets of points (segments):

- If $\delta \geq l_e$, $e \in I$;
- If $\delta < l_e$, let $n := \lceil \frac{l_e}{\delta} \rceil + 1$. We define $v_1 := v_a$, $v_n := v_b$ and $v_2, \ldots, v_{n-1} \in e$ such that $l(v_a, v_i) = (i-1)\frac{l_e}{n-1}$ for $i = 2, \ldots, n-1$. Then, $(v_i, v_{i+1}) \in I$ for all $i = 1, \ldots, n-1$.

We consider $N' = (V', E' = I)$, where V' contains the endpoints of I. The new network N' satisfies that $\delta \geq l_e$ for all $e \in E'$, and it is isomorphic to N with respect to the length function. Indeed, since N' is obtained by subdividing edges in N, $C(N) = C(N')$ and a set of points δ-covers N if and only it δ-covers N'. Therefore, Assumption 9.2 always holds after the network N is transformed into N' (via a preprocessing step). Such transformation yields a network with more nodes and edges, which has a direct impact on the size of optimization models. In Sect. 9.6, we present a model that avoids this effect.

9.2.1 Observations

In the following, we give several observations of optimal δ-covers, which guide our quantitative analysis of covering conditions and the resulting MILP model of Sect. 9.3.

Observation 9.3. *For an edge $e = (v_a, v_b) \in E$ and a point $p \in C(N)$, one of the following cases holds:*

1. *p cannot δ-cover any point in e;*
2. *p can δ-cover the whole e;*
3. *p can δ-cover a continuous portion of e containing either v_a or v_b;*
4. *p can δ-cover two continuous portions of e, which do not intersect, each contains either v_a or v_b.*

Observation 9.4. *(a similar statement was proven in [142]) There exists an optimal δ-cover that satisfies:*

i) Each edge $e \in E$ has at most two facilities (due to Assumption 9.2);

ii) If there are two facilities in the edge e, we can assume without loss of generality that they are located at end nodes v_a, v_b (this follows from i) and Assumption 9.2).

As a consequence, the set of candidate facilities of a δ-cover is in one-to-one correspondence to the edges and nodes of the network.

With the above observations, we can already give a high-level description of the covering characterization behind our model. Namely, if we fix a set of facilities on some network edges, there would be some edges completely covered regardless of the exact facility locations within their edges. Some other edges would be partially covered from the left-end node and/or from the right-end node, and how much depends on the actual facility locations. So we have variables that specify the exact facility locations on each edge. Finally, we stipulate that the cover from the left and the cover from the right better exceed the edge length. One difficulty is that the corresponding covering function is not linear However, we show that such a function is a piece-wise linear function, which can be modeled by a MILP. The aim of the remainder of this section is to give a mathematical specification of the above characterization.

9.2.2 Covering conditions

We give a sufficient and necessary condition that the network is δ-covered by installed facilities.

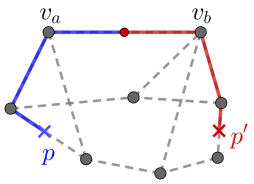
Proposition 9.5. *Let $\mathcal{P} = \{p_i\}_{p_i \in C(N)}$ be a finite set of points in $C(N)$. An edge $e = (v_a, v_b) \in E$ is δ-covered by $\mathcal{P}$ if and only if either there exists $p \in \mathcal{P} \cap e$ or*

$$\max\{\delta - \min_{p \in \mathcal{P}} d(v_a, p), 0\} + \max\{\delta - \min_{p \in \mathcal{P}} d(v_b, p), 0\} \geq l_e. \tag{9.2}$$

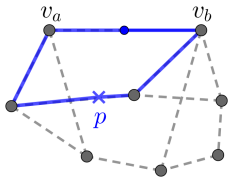
Moreover, the set $\mathcal{P}$ is a δ-cover of N if and only if for each $e \in E$, either there exists $p \in \mathcal{P} \cap e$ or (9.2) *is satisfied.*

Proof. If there exists $p \in \mathcal{P} \cap e$, then e is δ-covered by p due to Assumption 9.2. Otherwise, for each $i \in \{a, b\}$, let us consider $p_i^* \in \mathcal{P}$ such that $d(v_i, p_i^*) = \min_{p \in \mathcal{P}} d(v_i, p)$ and let $\pi_i^* \in \Pi^*(v_i, p_i^*)$ be a shortest path between v_i and p_i^*, i.e. $l_{\pi_i^*} = d(v_i, p_i^*)$. Condition (9.2) can be rewritten as follows:

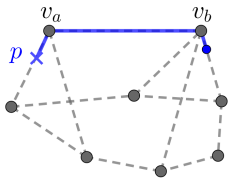
$$\max\{\delta - l_{\pi_a^*}, 0\} + \max\{\delta - l_{\pi_b^*}, 0\} \geq l_e.$$



(a) By two points



(b) By one point, through both ends



(c) By one point, through one end

Figure 9.3 Covering of an edge $e = (v_a, v_b) \in E$

Note that $\max\{\delta - l_{\pi_i^*}, 0\}$ represents the maximum length that can be δ-covered by $\mathcal{P}$ (specifically, from p_i^*) after passing through v_i. Since the path(s) that δ-cover e must contain v_a and/or v_b, the edge is covered if and only if these "maximum lengths" for v_a and v_b add up to more than l_e.

□

Fig. 9.3 illustrates Prop. 9.5. It shows three ways of covering the same edge $e = (v_a, v_b) \in E$ for a given network. The edges of the network are depicted with dashed lines, while the different paths through which e is covered are delimited with continuous bold traces. Facility locations are marked with the symbol 'x'. Fig. 9.3a depicts two facilities located at p, p' that cover two portions of the edge, which contain v_a and v_b respectively. In this case, the $\min$ functions inside (9.2) are attained respectively at p and p'. In the middle, Fig. 9.3b shows a single location p that covers e through two different paths, which traverse v_a and v_b respectively. These paths form a cycle that contains p and e. In this case, the two $\min$ operations inside (9.2) are attained at the same point, p. Finally, Fig. 9.3c illustrates the case in which a single facility located at p covers e through one of its end nodes, v_a. Here, one of the $\max$ operators in (9.2) is equal to zero (p is further from v_b than δ).

9.2.3 Covering delimitation and simplification

For an optimization or search problem, *delimitation* refers to the reduction of the candidate space. FDS is studied in related works as a way for reduction of CSCP (under some assumption), and hence it is a kind of delimitation. Instead of FDS, we consider a different delimitation that can be used for the general CSCP, and which allows us to obtain a reduced MILP formulation.

The characterization in Prop. 9.5 is based on the individual covering of every edge in the network. When considering possible locations to cover a fixed edge, we can restrict ourselves to its surroundings within the radius δ. Delimiting those parts of the network that could "contribute" to covering a particular edge or node reduces the search space. We then introduce three kinds of delimitation: *potential covers*, *complete covers* and *partial covers*. We will represent the covering condition under such delimitation. In effect, the covering condition

has a simplified form compared to its general form in Prop. 9.5, and an adequate preprocessing procedure can reduce and strengthen our MILP model.

We find that, for every node, there may exist potential covers, i.e., a set of edges and nodes where, if a facility is located, it can *possibly* δ-cover this node. The potential covers in the following definition delimit the edges and nodes of $C(N)$ that can contribute to covering a particular node of the network.

Definition 9.6. *For each $v \in V$, the potential covers of v are the candidate facility locations to cover v:*

$$\begin{aligned}
&\mathcal{E}(v) := \{e' = (v'_a, v'_b) \in E :\ d(v, v'_i) \leq \delta \text{ for some } i \in \{a, b\}\} \\
&\mathcal{V}(v) := \{v' \in V :\ d(v, v') \leq \delta\} \\
&\mathcal{F}(v) := \mathcal{E}(v) \cup \mathcal{V}(v).
\end{aligned}$$

Clearly, v is not reachable within the radius δ for any facility installed outside $\mathcal{F}(v)$. Regarding the covering of edges, an edge incident to v could be covered by some of the facilities in $\mathcal{F}(v)$.

We find that, for every edge e, there may exist complete covers, i.e., a set of edges and nodes where, if a facility is located, it can *always* δ-cover the edge e, regardless of the exact facility location. Once there is a facility within a complete cover, the whole edge e is guaranteed to be covered by this facility. The following definition serves to delimit the network to such complete covers.

Definition 9.7. *For each $e = (v_a, v_b) \in E$, the complete covers of e are the candidate facility locations that can completely cover e:*

$$\begin{aligned}
&\mathcal{E}_{\mathrm{c}}(e) := \{e' \in E : \forall p' \in e', \forall p \in e,\ d(p, p') \leq \delta\} \\
&\mathcal{V}_{\mathrm{c}}(e) := \{v' \in V : \forall p \in e,\ d(p, v') \leq \delta\} \\
&\mathcal{F}_{\mathrm{c}}(e) := \mathcal{E}_{\mathrm{c}}(e) \cup \mathcal{V}_{\mathrm{c}}(e).
\end{aligned} \tag{9.3}$$

If a facility is placed at $\mathcal{F}_{\mathrm{c}}(e)$ (either at a node in $\mathcal{V}_{\mathrm{c}}(e)$ or at a point on an edge belonging to $\mathcal{E}_{\mathrm{c}}(e)$), we can immediately conclude that e is δ-covered. Note that any facility placed at e' can completely cover e if and only if any facility placed at e can completely cover e'. That is, $\mathcal{E}_{\mathrm{c}}$ is symmetric over E. On the other hand, it is obvious that $e \in \mathcal{E}_{\mathrm{c}}(e)$, and $v_a, v_b \in \mathcal{V}_{\mathrm{c}}(e)$, for all $e = (v_a, v_b) \in E$.

Given a node v, we can characterize the complete covers of the incident edges to the node v. This helps us refine the potential covers of this node. The following definition identifies those candidate facility locations in the potential covers of the node v that cannot completely cover any incident edges to v.

Definition 9.8. *We define the following sets for each $v \in V$:*

$$\begin{aligned}
&\mathcal{E}_{\mathrm{p}}(v) := \{e' \in \mathcal{E}(v) :\ \exists e \in E(v),\ e' \notin \mathcal{E}_{\mathrm{c}}(e)\} \\
&\mathcal{V}_{\mathrm{p}}(v) := \{v' \in \mathcal{V}(v) :\ \exists e \in E(v),\ v' \notin \mathcal{V}_{\mathrm{c}}(e)\} \\
&\mathcal{F}_{\mathrm{p}}(v) := \mathcal{V}_{\mathrm{p}}(v) \cup \mathcal{E}_{\mathrm{p}}(v).
\end{aligned} \tag{9.4}$$

We call these sets the partial covers *of $E(v)$, where $E(v) := \{e \in E :\ v \in e\}$ is the set of incident edges to v.*

The set $\mathcal{F}_{\mathrm{p}}(v)$ contains those candidate locations that can contribute to partially covering some of the edges in $E(v)$. Note that, if a facility is placed at $\mathcal{F}(v) \setminus \mathcal{F}_{\mathrm{p}}(v)$, then this facility completely covers $E(v)$.

Definitions 9.6, 9.7, and 9.8 provide us with a refined covering condition. Indeed, the following proposition is a consequence of Prop. 9.5 and the aforementioned definitions, and will be used to characterize coverings in the MILP formulation presented in Sect. 9.3.

Proposition 9.9. *A finite set $\mathcal{P}$ of points in $C(N)$ is a δ-cover of N if and only if, for each $e = (v_a, v_b) \in E$, either $\mathcal{P} \cap \mathcal{F}_{\mathrm{c}}(e) \neq \varnothing$ or*

$$\sum_{i \in \{a,b\}} \max\left\{0, \delta - \min_{p \in \mathcal{P} \cap \mathcal{F}_{\mathrm{p}}(v_i)} d(v_i, p)\right\} \geq l_e. \tag{9.5}$$

Moreover,

$$\min_{p \in \mathcal{P} \cap \mathcal{F}_{\mathrm{p}}(v_i)} d(v_i, p) = \min\left\{ \min_{v' \in \mathcal{P} \cap \mathcal{V}_{\mathrm{p}}(v_i)} d(v_i, v'), \min_{p \in \mathcal{P} \cap (\mathcal{E}_{\mathrm{p}}(v_i) \setminus V)} d(v_i, p) \right\}, \text{ for } i = a, b.$$

Proof. For $i \in \{a, b\}$, the equality $\mathcal{F}(v_i) = \mathcal{F}_{\mathrm{p}}(v_i) \cup \mathcal{F}_{\mathrm{c}}(e)$ holds by definition, which gives the new necessary and sufficient covering condition.

For the second statement of the proposition, we have $\mathcal{F}_{\mathrm{p}}(v_i) = \mathcal{V}_{\mathrm{p}}(v_i) \cup \mathcal{E}_{\mathrm{p}}(v_i)$ from Definition 9.8. Then, it suffices to see that we can take $p \in \mathcal{P} \cap (\mathcal{E}_{\mathrm{p}}(v_i) \setminus V)$ in the second inner $\min$ operator of the right-hand side instead of $p \in \mathcal{P} \cap \mathcal{E}_{\mathrm{p}}(v_i)$. Let $e' = (v'_a, v'_b) \in \mathcal{E}_{\mathrm{p}}(v_i)$. We prove that the end nodes of e' can be excluded from the second inner $\min$ operator. Let $i' \in \{a, b\}$, we denote $\bar{i'} = b$ if $i' = a$ and $\bar{i'} = a$ if $i' = b$. First, by definition, $d(v_i, v'_{i'}) \leq \delta$ for some $i' \in \{a, b\}$. Therefore, $v'_{i'} \in \mathcal{V}_{\mathrm{p}}(v_i)$, and we can exclude it from the second inner $\min$ operator (this node is already considered by the first inner $\min$ operator). We consider now the other end node of e'. If $d(v_i, v'_{\bar{i'}}) \leq \delta$ then, similarly, $v'_{\bar{i'}}$ can be excluded from the second inner $\min$. Otherwise, we know that the outer $\min$ is not attained at $v'_{\bar{i'}}$, as $d(v_i, v'_{i'}) \leq \delta < d(v_i, v'_{\bar{i'}})$, thus $v'_{\bar{i'}}$ can be disregarded. □

Remark 9.10. *The left hand side function in* (9.5) *is a PWL function w.r.t. the distance d. Moreover, we cannot reformulate it into a system of linear constraints, because the left hand side function involves the max functions, which is convex w.r.t. its argument. In this chapter, we consider a big-M method to reformulate the PWL function as a MILP representable function.*

Remark 9.11. *The covering conditions described both in Propositions 9.5 and 9.9 would be also applicable if only a subset of E, $J \subseteq E$, is to be covered. Indeed, these covering conditions are based on the individual coverage of the edges, so it would be sufficient to apply them just to the edges in J. As a consequence, our methods, including the MILP formulation and algorithms presented in the next sections, apply to this more general version of the CSCP.*

In order to exploit the newly defined potential, complete, and partial covers in our formulation, from a practical viewpoint, we need to have some characterizations that can operate in a computer. Definition 9.6, which introduces potential covers, satisfies this requirement. Indeed, it just depends on distances between pairs of nodes, which we can easily calculate. Conversely, Definition 9.7 presents complete covers with a condition that must be satisfied by "the infinitely many points of an edge", which is not directly computable. Finally, the elements in the partial covers defined by Definition 9.8 can be easily calculated once both potential and complete covers are known.

9.2.4 Characterization of complete covers

In the following, we focus on characterizing the complete covers, which will be useful for our MILP formulation and tailored algorithms, (see forthcoming Sections 9.3 and 9.5). To begin with, we note that Prop. 9.5 already gives us a characterization of the nodes in $\mathcal{V}_{\mathrm{c}}(e)$. Indeed, it is easy to observe that, for a given $e \in E$, $v \in \mathcal{V}_{\mathrm{c}}(e)$ if and only if $\mathcal{P} := \{v\}$ δ-covers e. We then focus on the sets $\mathcal{E}_{\mathrm{c}}(e)$. On the one hand, it is clear that, for every edge $e = (v_a, v_b) \in E$,

$$\mathcal{E}_{\mathrm{c}}(e) \subseteq \mathcal{E}(v_a) \cap \mathcal{E}(v_b).$$

Moreover, if we define $\mathcal{E}_{\mathrm{c}}(v) \subseteq \mathcal{E}(v)$ as follows, we have a tighter set containing $\mathcal{E}_{\mathrm{c}}(e)$.

Definition 9.12. *The edges that can completely cover a node $v \in V$ are:*

$$\mathcal{E}_{\mathrm{c}}(v) := \{e' \in E : \forall p' \in e',\ d(v, p') \leq \delta\}.$$

It is clear that the following observation holds.

Observation 9.13. *For any $e = (v_a, v_b) \in E$, $\mathcal{E}_{\mathrm{c}}(e) \subseteq \mathcal{E}_{\mathrm{c}}(v_a) \cap \mathcal{E}_{\mathrm{c}}(v_b)$.*

With this observation, we can limit the search of $\mathcal{E}_{\mathrm{c}}(e)$ in the set $\mathcal{E}_{\mathrm{c}}(v_a) \cap \mathcal{E}_{\mathrm{c}}(v_b)$, as we will show in Sect. 9.5, the latter set is easy to compute.

We recall that Definition 9.12 is somewhat the inverse of Definition 9.7. That is, $e' \in \mathcal{E}_{\mathrm{c}}(v)$ if and only if $v \in \mathcal{V}_{\mathrm{c}}(e')$. We present a set of intermediate statements in Definition 9.14, Lemma 9.15, and Lemma 9.16, which allow us to describe the edges in the complete cover set $\mathcal{E}_{\mathrm{c}}(e)$ as the main result in Prop. 9.18.

Definition 9.14. *Let $v \in V$ be a node and $e' = (v'_a, v'_b) \in E$ be an edge. For all $q \in [0, l_{e'}]$, we define the following functions:*

$$d_v(q) := \min\{d(v, v'_a) + q, d(v, v'_b) + l_{e'} - q\}$$
$$r_v(q) := \max\{\delta - d_v(q), 0\},$$

and constant:

$$Q_{ve'} := (d(v, v'_b) + l_{e'} - d(v, v'_a))/2,$$

which satisfies the following equation:

$$d(v, v'_a) + Q_{ve'} = d(v, v'_b) + l_{e'} - Q_{ve'}.$$

The function $d_v(q)$ represents the distance between v and a point $p' \in e'$ such that $q = l(v'_a, p')$, where $l(v'_a, p')$ measures the length of the continuum $(v'_a, p') \subseteq e'$. The inner terms in the minimization that defines $d_v(q)$ coincide for $q = Q_{ve'}$. Informally, $Q_{ve'}$ is the "bottleneck" coordinate on e' (see original definition in [94]), for which the distance to v is the same if we go through v'_a or v'_b. Indeed, since $|d(v, v'_b) - d(v, v'_a)| \leq l_{e'}$, it follows that $0 \leq Q_{ve'} \leq l_{e'}$. Note that $d_v(q)$, $r_v(q)$, and $Q_{ve'}$ depend also on the edge e'.

Lemma 9.15. *Let $e = (v_a, v_b)$. An edge e' is in $\mathcal{E}_{\mathrm{c}}(e)$ if and only if $r_{v_a}(q) + r_{v_b}(q) \geq l_e$ for all $q \in [0, l_{e'}]$.*

Proof. $e' \in \mathcal{E}_{\mathrm{c}}(e)$ if and only e is δ-covered by any point $p' \in e'$. Take $\mathcal{P} = \{p'\}$ in Prop. 9.5, e is δ-covered by $\mathcal{P}$, if and only if

$$\max\{\delta - d(v_a, p'), 0\} + \max\{\delta - d(v_b, p'), 0\} \geq l_e$$

holds for all $p' \in e'$. Let $q = l(v'_a, p'),\ q \in [0, l_{e'}]$, be the measure of the sub-edge $(v'_a, p') \subseteq e'$. Then, the observation that $d_{v_i}(q) = d(v_i, p')$ for $i = \{a, b\}$ completes the proof. □

We present the following lemma without proof. The lemma is a direct consequence of Definition 9.14.

Lemma 9.16. *Let $v \in V$ and $e' = (v'_a, v'_b) \in E$. Then, $0 \leq Q_{ve'} \leq l_{e'}$, the function $d_v(q)$ is increasing when $q \in [0, Q_{ve'}]$, and it decreases for $q \in [Q_{ve'}, l_{e'}]$. Moreover, $d_v(q)$ admits the following piece-wise linear representation:*

$$d_v(q) = \begin{cases} d(v, v'_a) + q & \text{if } q \leq Q_{ve'}, \\ d(v, v'_b) + l_{e'} - q & \text{if } q \geq Q_{ve'}. \end{cases}$$

Note that with Obs. 9.13, to find $\mathcal{E}_{\mathrm{c}}(e)$, we can check Lemma 9.15 for $e' \in \mathcal{E}_{\mathrm{c}}(v_a) \cap \mathcal{E}_{\mathrm{c}}(v_b)$: if $r_{v_a}(q) + r_{v_b}(q) \geq l_e$, then e' is in $\mathcal{E}_{\mathrm{c}}(e)$. Then, we consider $v \in V$, and we want to characterize r_v for all points on edges $e' \in E$ such that $e' \in \mathcal{E}_{\mathrm{c}}(v)$. We present the following result without proof.

Lemma 9.17. *Given $v \in V$, and $e' = (v'_a, v'_b) \in \mathcal{E}_{\mathrm{c}}(v)$. If $e' \in \mathcal{E}_{\mathrm{c}}(v)$, then $r_v(q)$ admits the following piece-wise linear representation:*

$$r_v(q) = \begin{cases} \delta - (d(v, v'_a) + q) & \text{if } q \leq Q_{ve'}, \\ \delta - (d(v, v'_b) + l_{e'} - q) & \text{if } q \geq\ Q_{ve'}. \end{cases}$$

Finally, we have a tractable version of Lemma 9.15.

Proposition 9.18. *Let $e = (v_a, v_b)$. An edge $e' \in \mathcal{E}_{\mathrm{c}}(v_a) \cap \mathcal{E}_{\mathrm{c}}(v_b)$ is in $\mathcal{E}_{\mathrm{c}}(e)$ if and only if*

$$r(q) \geq l_e \quad \text{for all} \quad q = Q_{v_a e'}, Q_{v_b e'},$$

where $r(q) := r_{v_a}(q) + r_{v_b}(q)$.

Proof. From Lemma 9.15, $e' \in \mathcal{E}_{\mathrm{c}}(e)$ if and only if $\min_{q \in [0, l_{e'}]} r(q) \geq l_e$. Due to Lemma 9.16, the minimum argument must be some of the breakpoints in the piece-wise linear description of $r_v(q)$. Then, it suffices to check

$$\min\{r(q) :\ q \in \{0, Q_{v_a e'}, Q_{v_b e'}, l_{e'}\}\} \geq l_e.$$

Since $e' \in \mathcal{E}_{\mathrm{c}}(v_a) \cap \mathcal{E}_{\mathrm{c}}(v_b)$, $r_{v_i}(0) \geq l_e$ and $r_{v_i}(l_{e'}) \geq l_e$ always holds for all $i \in \{a, b\}$. It suffices thus to check the following condition

$$\min\{r(q) :\ q \in \{Q_{v_a e'}, Q_{v_b e'}\}\} \geq l_e.$$

Then, the result follows. □

9.3 MILP formulation

We present the variables of the MILP formulation first. For each $v \in V$, there is one candidate facility (fixed location); and for each $e \in E$, there is another one (location within the interior of e, $e \setminus \{v_a, v_b\}$). Then, the finite set $\mathcal{F} \coloneqq E \cup V$ will be used to index the candidate facilities. In our MILP formulation, there are two decisions associated with each $f \in \mathcal{F}$. One is to decide if a facility is installed at f. The second is only necessary for those facilities installed at the interior of edges, and consists in determining their locations within the corresponding edges.

To represent the first of the above decisions, we define the following binary variables, which we call the *placement variables*:

$$y_f = 1 \text{ if a facility is installed at } f, \quad \text{for all } f \in \mathcal{F}.$$

We identify the set of installed facilities with $\mathcal{F}_1 = \{f \in \mathcal{F} : y_f = 1\}$. To represent the second of the above decisions, we define the following continuous variables, which we name the *coordinate variables*:

$$q_e = \begin{cases} l(v_a, p) & \text{if } y_e = 1 \text{ and a facility is installed at } p \in e \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } e = (v_a, v_b) \in E.$$

We use v' and e' to denote nodes and edges where facilities are installed, and use v and e to denote nodes and edges to be covered, respectively. We refer to v_a, v_b as the end nodes of $e = (v_a, v_b) \in E$; given $e' = (v'_a, v'_b) \in E$, we refer to v'_a, v'_b as its end nodes.

We use the necessary and sufficient condition of δ-covering in Prop. 9.9, and the second result in this proposition regarding the distance function. Other than the placement and coordinate variables, some additional variables are used, which we present next.

- $\forall e \in E, w_e \in \{0, 1\}$, if $\mathcal{F}_{\mathrm{c}}(e) \cap \mathcal{F}_1 \neq \varnothing$, then $w_e = 1$;
- $\forall v \in V, r_v \in [0, +\infty)$;
- $\forall v \in V, x_v \in \{0, 1\}, = 1$, if $\forall e \in E(v)$, $\mathcal{F}_{\mathrm{c}}(e) \cap \mathcal{F}_1 \neq \varnothing$, then $x_v = 1$;
- $\forall v \in V$, $v' \in \mathcal{V}_{\mathrm{p}}(v), z_{vv'} \in \{0, 1\}$, if $\forall e = (u, v) \in E(v)$ s.t. $w_e = 0$, $\max\{0, \delta - d(v, v')\} + r_u \geq l_e$, then $z_{vv'} = 1$;
- $\forall v \in V$, $(e', i') \in \mathcal{EI}_{\mathrm{p}}(v)$, if $\forall e = (u, v) \in E(v)$ s.t. $w_e = 0$, $\max\{0, \delta - \tau_{ve'i'}(q_{e'})\} + r_u \geq l_e$, then $z_{ve'i'} \in \{0, 1\} = 1$.

where

$$\tau_{ve'i'}(q) \coloneqq d(v, v_{i'}) + \mathbf{1}_{i'=a} q + \mathbf{1}_{i'=b}(l_{e'} - q),$$

and

$$\mathcal{EI}_{\mathrm{p}}(v) \coloneqq \{(e' = (v'_a, v'_b), i') \in \mathcal{E}_{\mathrm{p}}(v) \times \{a, b\} : \ d(v, v'_{i'}) \leq \delta\}.$$

We sometimes refer to r_v as the "residual cover" at node v, since it represents the maximum remaining length that can be covered after reaching v from "a sufficiently close" facility. Remembering Defn. 9.14, these variables must satisfy:

$$r_v \leq r_v(q_e), \ \text{ for all } e \in \mathcal{F}_1 \cap (E \setminus E(v)).$$

In an optimal solution, it can be $r_v = r_v(q_e)$ for an edge $e \in E$ as stated above. In this case, r_v is the maximum remaining length that can be covered after reaching v from the closest facility. However, we do not impose this equality in our formulation, since it is enough for guaranteeing the coverage of $e = (v_a, v_b) \in E$ that the sum of the residuals $r_{v_a} + r_{v_b}$ exceeds l_e (see Prop. 9.9). That is, if l_e is already exceeded by $r_{v_a} + r_{v_b}$ for some $r_{v_i} < r_{v_i}(q_e)$, then the coverage condition of Prop. 9.9 will hold.

We denote by M_* the sufficiently large big-M constant associated with index $*$, the value of which will be determined later. Our formulation of the CSCP reads as follows:

$$\min \sum_{f \in \mathcal{F}} y_f \tag{9.6a}$$

$$\text{s.t. } w_e \geq y_f \qquad e \in E, f \in \mathcal{F}_{\mathrm{c}}(e) \tag{9.6b}$$

$$w_e \leq \sum_{f \in \mathcal{F}_{\mathrm{c}}(e)} y_f \qquad e \in E \tag{9.6c}$$

$$x_v \geq 1 - \sum_{e \in E(v)} (1 - w_e) \qquad v \in V \tag{9.6d}$$

$$x_v \leq w_e \qquad v \in V, e \in E(v) \tag{9.6e}$$

$$y_{v'_{i'}} + y_{e'} \leq 1 \qquad e' \in E, i' \in \{a, b\} \tag{9.6f}$$

$$q_{e'} \leq l_{e'} y_{e'} \qquad e' \in E \tag{9.6g}$$

$$l_e(1 - w_e) \leq r_{v_a} + r_{v_b} \qquad e \in E \tag{9.6h}$$

$$x_v + \sum_{v' \in \mathcal{V}_{\mathrm{p}}(v)} z_{vv'} + \sum_{(e',i') \in \mathcal{EI}_{\mathrm{p}}(v)} z_{ve'i'} = 1 \qquad v \in V \tag{9.6i}$$

$$z_{vv'} \leq y_{v'} \qquad v \in V, v' \in \mathcal{V}_{\mathrm{p}}(v) \tag{9.6j}$$

$$z_{ve'i'} \leq y_{e'} \qquad v \in V, (e', i') \in \mathcal{EI}_{\mathrm{p}}(v) \tag{9.6k}$$

$$r_v \leq M_v(1 - x_v) \qquad v \in V \tag{9.6l}$$

$$r_v \leq M_{vv'}(1 - z_{vv'}) + \delta - d(v, v') \qquad v \in V, v' \in \mathcal{V}_{\mathrm{p}}(v) \tag{9.6m}$$

$$r_v \leq M_{ve'i'}(1 - z_{ve'i'}) + \delta - \tau_{ve'i'}(q_{e'}) \qquad v \in V, (e', i') \in \mathcal{EI}_{\mathrm{p}}(v) \tag{9.6n}$$

$$y_f, w_e \in \{0, 1\} \qquad f \in \mathcal{F}, e \in E \tag{9.6o}$$

$$x_v, z_{vv'}, z_{ve'i'} \in \{0, 1\} \qquad v \in V, v' \in \mathcal{V}_{\mathrm{p}}(v), (e', i') \in \mathcal{EI}_{\mathrm{p}}(v) \tag{9.6p}$$

$$q_{e'}, r_v \geq 0 \qquad e' \in E, v \in V. \tag{9.6q}$$

Constraints (9.6b) and (9.6c) model the logic or constraint $w_e = \vee_{f \in \mathcal{F}_{\mathrm{c}}(e)} y_f$. Constraints (9.6d) and (9.6e) enforce the logic constraint $x_v = \wedge_{e \in E(v)} w_e$, that is, x_v is the product of the w_e variables such that $e \in E(v)$. Constraints (9.6f) prevent two facilities in a solution from being installed respectively at the interior of an edge and one of their end nodes. Constraints (9.6g) bound the coordinate variables with the corresponding edge length, and set them to zero if no facility is located at its interior. The covering condition in Prop. 9.9 is enforced by (9.6h). If $w_e = 1$, then the condition is satisfied (e is covered by $\mathcal{F}_{\mathrm{c}}(e)$). Otherwise, the inequality (9.5) of the proposition has to be satisfied. The rest of the constraints of the model (9.6i)-(9.6n), together with variables r, x, q, and z, aim at modeling (9.5). To begin with, (9.6i) impose that, for each $v \in V$, one of the following statements holds:

i) All incident edges to v, $e \in E(v)$, are completely covered by facilities placed at their complete covers, $\mathcal{F}_{\mathrm{c}}(e)$ ($w_e = 1$ for all $e \in E(v)$, $x_v = 1$).

ii) A sufficiently close facility to v is installed at $v' \in \mathcal{V}_{\mathrm{p}}(v)$ ($z_{vv'} = 1$), that is,

$$\max\{0, \delta - d(v, v')\} + r_u \geq l_e \qquad \forall u \in V \text{ s.t. } (u, v) = e \in E \text{ and } w_e = 0;$$

iii) A sufficiently close facility to v is installed at $e' \in \mathcal{E}_{\mathrm{p}}(v)$ and v is reached through $v'_{i'}$ of e' ($z_{ve'i'} = 1$), that is,

$$\max\{0, \delta - \tau_{ve'i'}(q_{e'})\} + r_u \geq l_e \qquad \forall u \in V \text{ s.t. } (u, v) = e \in E \text{ and } w_e = 0.$$

If the case i) above holds, then the covering condition in (9.6h) is satisfied for all $e \in E(v)$, regardless of the value of the residual cover variables. Otherwise, suppose that $x_v = 0$ and $w_e = 0$ for some $e \in E(v)$. In this case, the corresponding constraint (9.6h) is "active", that is, the inequality (9.5) of Prop. 9.9 has to be satisfied for e. Since $x_v = 0$, constraints (9.6i) impose that there is a facility among those installed at $\mathcal{F}_{\mathrm{p}}(v)$ that is sufficiently close one to v. This facility is the one bounding the residual variables r_v (see constraints (9.6m)-(9.6n)), which represent the terms in the left-hand side of (9.5). Constraints (9.6j) (resp. (9.6k)) ensure that $z_{vv'}$ (resp. $z_{ve'i'}$) can be one only if facility is installed at v' (resp. e'). Due to (9.6i), for every fixed node $v \in V$, at most one of the constraints in (9.6l)-(9.6n) will be active. If $x_v = 1$, (9.6l) enforces $r_v = 0$. Indeed, all the covering conditions (9.6h) are "inactive" and r_v is not needed to guarantee the coverage of any $e \in E(v)$. Otherwise, if $x_v = 0$, (9.6l) reads $r_v \leq M_v$, where M_v is a big-enough constant that does not restrict the value of the residual. Finally, constraints (9.6m)-(9.6n) bound r_v by $\delta - d(v, p) \geq 0$ for a sufficiently close facility to v installed at p, when $x_v = 0$. The constants $M_{vv'}$ and $M_{ve'i'}$ are assumed to be big enough so that the constraints in (9.6m)-(9.6n) do not add anything to the model if $z_{vv'}$ or $z_{ve'i'}$ are zero, respectively. For instance, $M_v = M_{vv'} = \delta$ and $M_{ve'i'} = \delta + l_{e'}$ are valid values for these constants (we recall Assumption 9.2). Sect. 9.4 presents refined values of these big-Ms.

We observe that the number of variables and constraints in (9.6) can be reduced. Namely, for each $v \in V$ and $e' = (v'_a, v'_b) \in \mathcal{E}_{\mathrm{p}}(v)$, if $d(v, v'_a) + l_{e'} \leq d(v, v'_b)$ then $d(v, p) = d(v, v'_a) + l(v'_a, p)$ for every $p \in e'$. Similarly, if $d(v, v'_b) + l_{e'} \leq d(v, v'_a)$ then $d(v, p) = d(v, v'_b) + l(v'_b, p)$ always holds for all $p \in e'$. For such nodes and candidate facilities, we do not need both variables, $z_{ve'a}$ and $z_{ve'b}$, and corresponding constraints in (9.6n) (we know beforehand that one of these constraints would never be active if a facility is located at e'). Therefore, $\mathcal{EI}_{\mathrm{p}}(v)$ would only contain one of the pairs (e', a) or (e', b).

9.3.1 Comparative insights with respect to an existing MILP

To the best of our knowledge, the only existing MILP for the CSCP was proposed in [142]. The authors used a similar observation to ours with respect to optimal δ-covers. They noted that every edge contains at most one facility. Indeed, in their setting, if two facilities are located at both end-nodes of an edge $e = (v_a, v_b)$ one of them is considered to be "hosted" by an adjacent edge, $e' \in E(v_a) \cup E(v_b)$ (by optimality, neither v_a nor v_b is a leaf). Their location variables are indexed then by E.

However, this approach has issues with symmetry, which leads to redundant solutions. Indeed, many different solutions to the MILP model represent the same facility locations to the CSCP, since there exist many combinations of the edges "hosting" the facilities that are located at nodes. Let us consider an example: let $v \in V$ be a node, $e \in E(v)$ be an incident edge, and

	Variables		Constraints
	Binaries	Continuous	
MILP (9.6)	$\|V\|^2 + 2(\|V\|\|E\| + \|V\| + \|E\|)$	$\|V\| + \|E\|$	$\|E\|^2 + \|V\|^2 + 5\|E\|\|V\| + 7\|E\| + 3\|V\|$
MILP in [142]	$\|E\|^3 + 3\|V\|\|E\| + \|E\|$	$3\|V\|\|E\| + \|E\|$	$3\|E\|^3 + 8\|E\|\|V\| + \|E\|$

Table 9.1 Comparative summary on MILP formulations for the CSCP

$p \in e$ be a point on e. Even though the distance $d(p, v)$ is equal to ϵ for a very small $\epsilon > 0$, the point p is still located at e. However, when $d(p, v) = 0$, the point p is located at the node v and, consequently, at every edge in $E(v)$. When a facility is located at a node, the discontinuity there leads to the question: which node or edge do we choose to represent this facility? In our MILP model, we have a specific node facility variable (i.e., y_v) which prevents edge facility variables (i.e., $y_e, e \in E(v)$) from "hosting" facilities at v, thanks to constraint (9.6f).

On the other hand, we consider model size in terms of the number of variables. A second main difference between MILP (9.6) and the MILP in [142] is that the latter uses binary variables to identify the two edges containing the facilities that cover a given edge. On the one hand, this yields variables and constraints of $\mathcal{O}(|E|^3)$. On the other hand, multiple equivalent solutions arise when the edge in question can be covered by a single facility, as the authors commented themselves. Finally, the covering constraints in both formulations actually correspond to the same characterization of δ-cover, but are modeled in a slightly different way. Namely, the authors of [142] defined the "residual covers" for each edge (where a facility might be placed) and node of the network. Interested readers might consult [142] and the MILP therein, which we do not reproduce here for the sake of concision.

Nonetheless, Table 9.1 shows a comparative summary of the two formulations, based on the number of variables and constraints. This summary considers an upper bound on the size of MILP (9.6). That is, we take $\mathcal{F}_{\mathrm{c}}(e) = \mathcal{F}$, $\mathcal{V}_{\mathrm{p}}(v) = V$, and $\mathcal{E}_{\mathrm{p}}(v) = E$ for all $v \in V$ and $e \in E$—however, this would never be the case, as the partial and complete covers are complementary. On the other hand, Table 9.1 considers the MILP in [142] with $J = E$ (the set of edges to be covered).

9.4 Strengthening

In this section, we analyze modifications of the MILP (9.6) that can yield a tighter linear relaxation of this formulation. Namely, we tighten our big-M constraints (9.6l)-(9.6n) by devising small constants M_v, $M_{vv'}$, $M_{ve'i'}$, $\delta_{vv'}$, and $\delta_{ve'i'}$. We also present several families of valid inequalities. Valid inequalities define conditions that have to be satisfied by any feasible solution, and yield tighter linear programming relaxations, see e.g. [306].

9.4.1 Constants tightening

From the MILP formulation, it is easy to yield the following observation. For $v \in V$, it suffices for a facility $f \in \mathcal{F}_{\mathrm{p}}(v)$ to contribute to the residual cover r_v at most $U_v := \max_{e \in E(v)} l_e$. Indeed, r_v aims at ensuring that the inequality (9.5) of Prop. 9.9 is satisfied for all $e \in E(v)$. We define

$\delta_{vv'} = \min\{U_v + d(v, v'), \delta\}$, for $v' \in \mathcal{V}_{\mathrm{p}}(v)$,
$\delta_{ve'i'} = \min\{U_v + \max_{q\in[0,l_{e'}]} \tau_{ve'i'}(q), \delta\} = \min\{U_v + d(v, v'_{i'}) + l_{e'}\delta\}$, for $(e' = (v'_a, v'_b), i') \in \mathcal{EI}_{\mathrm{p}}(v)$.

Since U_v is a valid upper bound for the residual cover variable r_v, the big-Ms in the constraints (9.6m) and (9.6n) should guarantee that

$$M_{vv'} + \delta_{vv'} - d(v, v') \geq U_v,$$
$$M_{ve'i'} + \min_{q\in l_{e'}}(\delta_{ve'i'} - \tau_{ve'i'}(q)) \geq U_v.$$

Taking the minimums of the above big-Ms, we can now tighten the big-M constants of the MILP (9.6) as follows:

$$\begin{aligned}
M_v :=& U_v \\
M_{vv'} :=& U_v - (\delta_{vv'} - d(v, v')) = \max\{0, U_v + d(v, v') - \delta\} \\
M_{ve'i'} :=& U_v - \min_{q\in l_{e'}}(\delta_{ve'i'} - \tau_{ve'i'}(q)) = U_v - \delta_{ve'i'} + \max_{q\in l_{e'}} \tau_{ve'i'}(q) \\
=& U_v - \delta_{ve'i'} + d(v_i, v'_{i'}) + l_{e'} = \max\{0, U_v + d(v, v'_{i'}) + l_{e'} - \delta\},
\end{aligned}$$

where the last equations in the definition of $M_{vv'}$ and $M_{ve'i'}$ follow from the definition of $\delta_{vv'}$ and $\delta_{ve'i'}$, respectively.

Consequently, the constraints (9.6m) and (9.6n) should be replaced by:

$$r_v \leq M_{vv'}(1 - z_{vv'}) + \delta_{vv'} - d(v, v') \qquad v \in V, v' \in \mathcal{V}_{\mathrm{p}}(v) \tag{9.7a}$$
$$r_v \leq M_{ve'i'}(1 - z_{ve'i'}) + \delta_{ve'i'} - \tau_{ve'i'}(q_{e'}) \qquad v \in V, (e', i') \in \mathcal{EI}_{\mathrm{p}}(v). \tag{9.7b}$$

9.4.2 Valid inequalities

"Leafs" inequalities

If a node $v \in V$ has degree one, we can assume without loss of generality that no facility is located at v nor at its incident edge. Indeed, an equivalent δ-cover could be built by just moving such a facility to the unique neighbor of v in N. More than valid inequalities, the following are valid variable elimination:

$$y_v = 0;\ y_e = 0 \quad \forall v \in V \text{ s.t. } deg(v) = 1,\ e \in E(v). \tag{9.8}$$

"Adjacent edges" inequalities

Consider a node $v \in V$ of degree two. If there is a facility at v, then no facility is placed at the edges incident to v (we recall the model constraints (9.6f)). Otherwise, we can assume that at most one facility is placed at these edges in an optimal solution, which can be enforced by the following valid inequalities:

$$y_e + y_{e'} + y_v \leq 1 \quad \forall e, e' \in E,\ e \neq e',\ s.t.\ e \cap e' = v \text{ and } deg(v) = 2. \tag{9.9}$$

Fig. 9.4 illustrates the above inequalities. Fig. 9.4a shows the case in which a facility is located

(a) A facility is located at v (b) Solution with two facilities

Figure 9.4 Illustration of valid inequalities (9.9)

at v. Otherwise, if two facilities are placed at e and e' respectively, we can build an equivalent solution by moving one of these facilities to the end node of the corresponding edge that is not v, as depicted in Fig. 9.4b. We recall that the last statement holds due to our Assumption 9.2.

"Neighborhood" inequalities

Let us now consider a node $v \in V$, and suppose that there are several facilities placed at different edges in $E(v)$ in a feasible solution. Take $e^* \in E(v)$ containing a facility f^* such that $d(f^*, v) = \min\{d(f, v) : f \text{ is installed at } e \in E(v)\}$. The following proposition gives an equivalent feasible solution where the facilities at the edges $e \in E(v)$ such that $e \neq e^*$ are moved to the nodes.

Proposition 9.19. *Given a node $v \in V$ and an edge $e^* \in E(v)$, for any feasible solution $\hat{y}$ with several facilities placed at edges in $E(v)$, the following solution y is feasible and $\sum_{f\in\mathcal{F}} y_f \leq \sum_{f\in\mathcal{F}} \hat{y}_f$:*

- $y_u = 1$ *for all* $u \in V$ *such that* $e = (u, v) \in E(v)$, $e \neq e^*$, *and* $\hat{y}_e = 1$;
- $y_e = 0$ *for all* $e \in E(v)$ *such that* $e \neq e^*$, *and* $\hat{y}_e = 1$;
- $y_f = \hat{y}_f$ *otherwise.*

Proof. We denote by $N(v)$ the set of vertices adjacent to v. Consider the change of facilities from $\hat{y}$ to y. The facilities in the edges $E(v) \setminus \{e^*\}$ are 'pushed' to the vertices $N(v)$. For $u \in N(v)$, if there already exists a facility at u, and there is another facility 'pushed' to u, then these two facilities merge and they are accounted as one facility in y. Hence, the number of facilities of the solution y is at most that of the solution $\hat{y}$.

The proof then consists in showing that y is feasible. We will show that all edges are covered. Let us consider $e \in E$. If e was covered in $\hat{y}$ by facilities not placed at edges in $E(v)$ then it is still covered by these facilities in y. Suppose then that a facility placed at $e' \in E(v)$ with $e \neq e^*$ was covering e (or part of e) in solution $\hat{y}$, and let $e' = (u, v)$. We distinguish two cases. First, if the facility at e' was partially covering e through node u, then it clearly covers at least the same part of e in the new solution y (where the facility is moved to u). Otherwise, suppose the facility at e' was partially covering e through node v. In this case, the facility at e^* covers at least the same part of e (it is closer to v). Since this facility remains unchanged in the new solution, we can guarantee that e is still covered. □

As a consequence of Prop. 9.19, the following inequalities are valid:

$$\sum_{e\in E(v)} y_e \leq 1 - y_v \qquad \forall v \in V. \tag{9.10}$$

(a) A facility is located at v (b) Solution with several facilities

Figure 9.5 Illustration of valid inequalities (9.10)

Fig. 9.5 illustrates the valid inequalities (9.10). In particular, Fig. 9.5b illustrates the equivalent solution given in Prop. 9.19. It is easy to observe that these new inequalities are a generalization of inequalities (9.9). Moreover, constraints (9.10) dominate the model constraints (9.6f)—and are fewer.

9.5 Network processing

The network processing algorithm analyzes the network N to compute the parameters and sets needed to construct the MILP model (9.6), which we recall next:

1. $\mathcal{V}_{\rm c}(e)$, $\mathcal{E}_{\rm c}(e)$ for all edges $e \in E$;
2. $\mathcal{V}_{\rm p}(v), \mathcal{E}_{\rm p}(v), \mathcal{EI}_{\rm p}(v)$ for all nodes $v \in V$;
3. $d(v, v')$ for all pairs of nodes $v, v' \in V$ such that $d(v, v') \leq \delta$.

The above data is computed by Algorithms 9.1, 9.2 and 9.3. Algorithms 9.1 and 9.2 contain auxiliary functions, which are called within the main Algorithm 9.3. Algorithm 9.1 computes the sets $\mathcal{E}(v)$ and $\mathcal{V}(v)$ (which are not directly used in the MILP but necessary to obtain $\mathcal{E}_{\rm p}(v)$ and $\mathcal{V}_{\rm p}(v)$), and the distances $d(v, v')$ for all $v, v' \in V$ such that $d(v, v') \leq \delta$. Algorithm 9.1 also computes the sets $\mathcal{E}_{\rm c}(v)$, which will serve as intermediate sets to finally obtain $\mathcal{E}_{\rm c}(e)$ in Algorithm 9.3. The main task in Algorithm 9.3 is to compute the sets $\mathcal{V}_{\rm c}(e)$ and $\mathcal{E}_{\rm c}(e)$. To that aim, this algorithm calls both Algorithm 9.1 and the procedure "mutual" described in Algorithm 9.2. Once $\mathcal{V}_{\rm c}(e)$ and $\mathcal{E}_{\rm c}(e)$ are known, the computation of $\mathcal{V}_{\rm p}(v)$ and $\mathcal{E}_{\rm p}(v)$ in Algorithm 9.3 easily follows by definition.

In the following, we present Algorithm 9.1, which defines the function "nodeCover(N, δ, s)". This function, for each source node $s \in V$, outputs: $\mathcal{E}_{\rm c}(s)$, $\mathcal{E}(s)$, $\mathcal{V}(s)$, and $d(s, v)$ for all $v \in V$ such that $d(s, v) \leq \delta$ (otherwise the algorithm outputs $d(s, v) = +\infty$). The algorithm starts with empty sets $\mathcal{E}_{\rm c}(s), \mathcal{E}(s), U(s), \mathcal{V}(s)$, where $U(s)$ is used for intermediate calculations. The set Q denotes nodes whose shortest path (and distance) to s are unknown, and it is initialized to V. In the course of the algorithm, Q decreases, while $\mathcal{V}(s)$ increases. In Lines 7-11, the distance $d(s, v)$ and predecessor values $\mathrm{prev}_s(v)$ are initialized, for all $v \in V$. The while loop is an adaptation of the classic Dijkstra algorithm. Line 14 selects the node u with the shortest distance to s among all unprocessed nodes, and removes it from Q. If $d(s, u) > \delta$, then none of the remaining nodes in Q are reachable from s, and the search is pruned. Otherwise, the neighbors of u that are still in Q are inspected. For each $v \in Q \cap E(u)$, the edge (u, v) is first added to $\mathcal{E}(s)$. Then, the algorithm computes the length ℓ of a path from s to v that traverses u. If $\ell < d(s, v)$, then the distance and the predecessor for node v are updated in Lines 24- 25. In addition, if $\ell < \delta$, node v and edge (u, v) are added to $\mathcal{E}_{\rm c}(s)$ and $\mathcal{V}(s)$ in Lines 27 and 28,

respectively. Otherwise, the edge e is added to the undetermined set $U(s)$. Whether this edge belongs or not to the complete cover set $\mathcal{E}_{\rm c}(s)$ is decided later on in the algorithm. Namely, edges $e = (v_a, v_b) \in U(s)$ are processed in Lines 35-39: if e can be jointly δ-covered by s from two sides, then e is added to the complete cover $\mathcal{E}_{\rm c}(s)$.

Algorithm 9.2 describes the procedure "mutual", which determines, given $e = (v_a, v_b) \in E$ and a candidate edge for the complete cover $e' \in \mathcal{E}_{\rm c}(v_a) \cap \mathcal{E}_{\rm c}(v_b)$, whether $e' \in \mathcal{E}_{\rm c}(e)$. This algorithm is based on Prop. 9.18 in Sect. 9.2.

Network processing Algorithm 9.3 computes all the sets that are needed by the MILP formulation. The algorithm starts with empty sets $\mathcal{E}_{\rm c}(e), \mathcal{V}_{\rm c}(e), \mathcal{E}_{\rm p}(v), \mathcal{V}_{\rm p}(v)$, for $e \in E$ and $v \in V$. In Line 3, the algorithm loops through all nodes $v \in V$ and computes the function "nodeCover(N, δ, v)", storing its output. Then, the algorithm calculates the sets $\mathcal{V}_{\rm c}(e)$ for $e \in E$, by applying the symmetric relation between these sets and the sets $\mathcal{E}_{\rm c}(v)$ from "nodeCover(N, δ, v)". After that, in Line 10, the algorithm loops through all edges $e = (v_a, v_b) \in E$. It checks whether there is an edge $e' \in \mathcal{E}_{\rm c}(v_a) \cap \mathcal{E}_{\rm c}(v_b)$ such that $e' \in \mathcal{E}_{\rm c}(e)$ (equivalently, $e \in \mathcal{E}_{\rm c}(e')$) by calling the procedure "mutual". Since $e' \in \mathcal{E}_{\rm c}(e)$ if and only if $e \in \mathcal{E}_{\rm c}(e')$, the loop only runs over pairs such that $e < e'$ (we assume a total order on the elements of E). The loop starting in line 18, iterates on each node $v \in V$ and looks for $v' \in \mathcal{V}(v)$ such that there exists an $e \in E(v)$ but $e \notin \mathcal{E}_{\rm c}(v')$. The nodes v' found are added to $\mathcal{V}_{\rm p}(v)$. Finally, the loop in line 28 also iterates on $v \in V$, and looks for $e' = (v'_a, v'_b) \in \mathcal{E}(v)$ such that there exists $e \in E(v)$ but $e \notin \mathcal{E}_{\rm c}(e')$. Each edge found is added to $\mathcal{E}_{\rm p}(v)$, and, right after that, the set $\mathcal{EI}_{\rm p}(v)$ may be updated after checking the dominance rule described at the end of Sect. 9.3. We have the following complexity result for Algorithm 9.3.

Proposition 9.20. *Let D be an upper bound on the degree of the nodes of a connected network $N = (V, E)$. The time complexity of the network processing Algorithm 9.3 is $\mathcal{O}(|E|^2 + |V||E|(D + \log|V|))$.*

Proof. We first analyze the time complexity of the procedure nodeCover described in Algorithm 9.1. The main **while** loop is a modification of the Dijkstra algorithm, and it can be implemented with time complexity $\mathcal{O}((|E| + |V|)\log|V|)$, see [107]. Therefore, the overall time complexity of nodeCover is also $\mathcal{O}((|E| + |V|)\log|V|)$.

The network processing Algorithm 9.3 has four outer loops, and next we analyze the complexity of each outer loop. The first outer loop runs the nodeCover algorithm over the nodes, so its complexity is $\mathcal{O}(|V|(|E| + |V|)\log|V|)$. The second outer loop runs the mutual algorithm over the edges pairs, since the mutual algorithm has a constant time complexity, so the complexity of this loop is $\mathcal{O}(|E|^2)$. The third outer loop is composed of three **for** loops, and its time complexity is $\mathcal{O}(D|V|^2)$, which has an upper bound $\mathcal{O}(D|E||V|)$. The last outer loop is composed of three **for** loops, and its time complexity is $\mathcal{O}(D|E||V|)$. After summing up the complexity of these loops, we have that the total time complexity of Algorithm 9.3 is $\mathcal{O}(|V|((|E| + |V|)\log|V|) + |E|^2 + |V||E|D)$, or, equivalently, $\mathcal{O}(|V||E|\log|V| + |E|^2 + |V||E|D)$. □

Algorithm 9.1: single node δ-cover algorithm: nodeCover

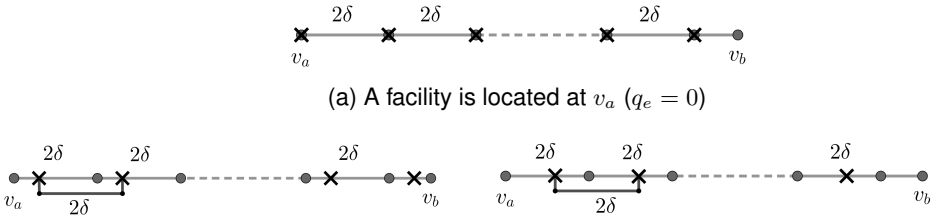
Input: Network $N = (V, E, ||)$, cover range $\delta > 0$, a source $s \in V$;
Output: $\mathcal{E}_c(s)$, $\mathcal{E}(s)$, $\mathcal{V}(s)$, $d(s, v)$ for all $v \in V$ (returns $d(s, v) = +\infty$ if $d(s, v) > \delta$);
Initialize set $Q \leftarrow V$;
Initialize sets $\mathcal{E}_c(s) \leftarrow \varnothing$, $\mathcal{E}(s) \leftarrow \varnothing$, $U(s) \leftarrow \varnothing$;
Initialize set $\mathcal{V}(s) \leftarrow \varnothing$;
for *each node* $v \in V$ **do**
    $d(s, v) \leftarrow +\infty$; ▷ Unknown distance from s to v
    $\text{prev}_s(v) \leftarrow \{\varnothing\}$; ▷ Unknown predecessor of v
end
$d(s, s) \leftarrow 0$;
add s to $\mathcal{V}(s)$;
while Q *is not empty* **do**
    $u \leftarrow \text{argmin}_{v \in Q}\, d(s, v)$;
    remove u from Q ; ▷ Take the closest node u and remove it from Q
    if $d(s, u) > \delta$ **then**
        $d(s, v) \leftarrow +\infty$ for all $v \in Q$; ▷ End of Dijkstra (all nodes in Q are outside the covering radius)
        break
    end
    for *each* $v \in Q$ *s.t.* $v \in E(u)$ **do**
        $e \leftarrow (u, v)$;
        add e to $\mathcal{E}(s)$; ▷ Edge e is in the potential cover set of s
        $\ell \leftarrow d(s, u) + l_e$; ▷ Path from s to v that traverses u
        if $\ell < d(s, v)$ **then**
            $d(s, v) \leftarrow \ell$; ▷ Update the distance to v
            $\text{prev}_s(v) \leftarrow u$; ▷ Update the predecessor of v
            if $\ell \leq \delta$ **then**
                add e to $\mathcal{E}_c(s)$; ▷ Edge e is in the complete cover set of s
                add v to $\mathcal{V}(s)$; ▷ Node v is in the potential cover set of s
            else
                add e to $U(s)$; ▷ Undetermined edge
            end
        end
    end
end
for *each edge* $e = (v_a, v_b)$ *in* $U(s)$ **do**
    if $v_a \in \mathcal{V}(s)$ *and* $v_b \in \mathcal{V}(s)$ *and* $\delta - d(s, v_a) + \delta - d(s, v_b) \geq l_e$ **then**
        add e to $\mathcal{E}_c(s)$; ▷ Edge e is completely covered
    end
end

Algorithm 9.2: Edge mutual cover algorithm: mutual

Input: Edges $e = (v_a, v_b), e' = (v'_a, v'_b)$ such that $e' \in \mathcal{E}_{\mathrm{c}}(v_a)$ and $e \in \mathcal{E}_{\mathrm{c}}(v_b)$.
Output: Boolean value indicating whether $e' \in \mathcal{E}_{\mathrm{c}}(e)$.
for $i \in \{a, b\}$ **do**
    $Q_{v_i e'} \leftarrow \frac{d(v_i, v'_b) + l_{e'} - d(v_i, v'_a)}{2}$;
end
for $i \in \{a, b\}$ **do**
    if $q \leq Q_{v_i e'}$ **then**
        $r_{v_i}(q) = \delta - (d(v_i, v'_a) + q)$;
    else
        $r_{v_i}(q) = \delta - (d(v_i, v'_b) + l_{e'} - q)$;
    end
end
if $r_{v_a}(Q_{v_a e'}) + r_{v_b}(Q_{v_a e'}) \geq e$ AND $r_{v_a}(Q_{v_b e'}) + r_{v_b}(Q_{v_b e'}) \geq e$ **then**
    return TRUE;
else
    return FALSE;
end

Algorithm 9.3: Network processing algorithm

Input: Network $N = (V, E, ||)$ and cover range $\delta > 0$;
Output: $\mathcal{E}_{\mathrm{c}}(e)$, $\mathcal{V}_{\mathrm{c}}(e)$, $\mathcal{E}_{\mathrm{p}}(v)$, $\mathcal{V}_{\mathrm{p}}(v)$, $\mathcal{EI}_{\mathrm{p}}(v)$, for all $e \in E$ and $v \in V$, and distance function d;
for *each node* $v \in V$; ▷ Computation of node complete covers $\mathcal{V}_{\mathrm{c}}(e)$
do
  $\mathcal{E}_{\mathrm{c}}(v), \mathcal{E}(v), \mathcal{V}(v), d(v, \cdot) \leftarrow$ nodeCover(N, δ, v) ;
  for *each edge* $e \in \mathcal{E}_{\mathrm{c}}(v)$ **do**
    add v to $\mathcal{V}_{\mathrm{c}}(e)$;
  end
end
for *each edge* $e = (v_a, v_b) \in E$; ▷ Computation of edge complete covers $\mathcal{E}_{\mathrm{c}}(e)$
do
  for *each edge* $e' = (v'_a, v'_b) \in E$, $e < e'$, *such that* $e' \in \mathcal{E}_{\mathrm{c}}(v_a) \cap \mathcal{E}_{\mathrm{c}}(v_b)$ **do**
    if *mutual*(e, e', d) **then**
      add e' to $\mathcal{E}_{\mathrm{c}}(e)$;
      add e to $\mathcal{E}_{\mathrm{c}}(e')$;
  end
end
for *each node* $v \in V$; ▷ Computation of node partial covers $\mathcal{V}_{\mathrm{p}}(v)$
do
  for *each node* $v' \in \mathcal{V}(v)$ **do**
    for *all* $e \in E(v)$ **do**
      if $e \notin \mathcal{E}_{\mathrm{c}}(v')$ **then**
        add v' to $\mathcal{V}_{\mathrm{p}}(v)$;
        break
    end
  end
end
for *each node* $v \in V$; ▷ Computation of edge partial covers $\mathcal{E}_{\mathrm{p}}(v)$ and $\mathcal{EI}_{\mathrm{p}}(v)$
do
  for *each edge* $e' = (v'_a, v'_b) \in \mathcal{E}(v)$ **do**
    for *all* $e \in E(v)$ **do**
      if $e' \notin \mathcal{E}_{\mathrm{c}}(e)$ **then**
        add e' to $\mathcal{E}_{\mathrm{p}}(v)$;
        if $d(v, v'_a) \leq \delta$ AND $d(v, v'_a) \leq d(v, v'_b) + l_{e'}$ **then**
          add (e', a) to $\mathcal{EI}_{\mathrm{p}}(v)$;
        if $d(v, v'_b) \leq \delta$ AND $d(v, v'_b) \leq d(v, v'_a) + l_{e'}$ **then**
          add (e', b) to $\mathcal{EI}_{\mathrm{p}}(v)$;
        break
    end
  end
end



(a) A facility is located at v_a ($q_e = 0$)

(b) A facility is located at the tail ($0 < q_e \leq \hat{l}_e$)

(c) No facility is located at the tail ($\hat{l}_e < q_e \leq 2\delta$)

Figure 9.6 Illustration of Prop. 9.21 for a long edge e

9.6 A reduced formulation for networks with long edges

The MILP (9.6) assumes $l_e \leq \delta$ for all $e \in E$, however, in many real-world networks, some edge lengths are greater than the covering radius. To reuse the previous results, one approach is to transform a network with long edges into another network with edge lengths at most δ. The transformation is by subdividing edges of the original network into smaller pieces, so the optimal cover does not change. This transformation enables us to apply MILP (9.6) or the MILP in [142] on the transformed network. We note that, in [142], there is a recursive transformation. However, the recursive transformation is not constructive, so we cannot employ it in practice. Another trivial transformation is by subdividing edges as suggested in Sect. 9.2. However, this strategy is a trivial heuristic, and it increases the number of edges and nodes of the transformed network, and thus the number of variables and constraints of the MILP model by a nonlinear factor.

In this section, we present an alternative approach to tackle networks with edge lengths greater than the covering radius. Instead of transforming the network, this approach directly treats long edges in the formulation by using specific sets of constraints and variables. We highlight that the approach is also applicable to "long paths". That is, if there is a path in the network whose intermediate nodes all have degree two, we can represent it by a single edge of length equal to the total length of the path. Indeed, the CSCP does not change after this transformation.

The main idea of the reduced formulation is to assume a predefined covering of those edges that are long enough. Such covering consists in placing facilities every 2δ distance units on the long edge. Let us consider $e \in E$ such that $l_e > 2\delta$. An edge satisfying this condition is called a *long edge*. We denote by $\hat{l}_e := l_e - 2\delta\lfloor l_e/(2\delta)\rfloor$ the length of the last piece of e after dividing it into pieces of measure 2δ. We call $\hat{l}_e$ the *tail* of e. The following proposition guarantees the correctness of the reduced formulation.

Proposition 9.21. *Let N be an undirected network, $e = (v_a, v_b) \in E$ be a long edge, and $\mathcal{P}'$ be a feasible δ-cover of N. Define $\mathcal{P}$ with $p \in \mathcal{P}$ for all $p \in \mathcal{P}' \setminus e$. Let $p_e \in \mathcal{P}' \cap e$ be such that $l(v_a, p_e) = \min_{p' \in \mathcal{P}' \cap e} l(v_a, p')$, and let $q_e := l(v_a, p_e)$ (here q_e represents a length, although it will also be a variable of the reduced MILP that we introduce afterward). Note that $q_e \in [0, 2\delta]$ (otherwise, $\mathcal{P}'$ would not be a δ-cover). The set $\mathcal{P}$ can be completed in such a way that it δ-covers N and $|\mathcal{P}| \leq |\mathcal{P}'|$, as follows:*

- *$p_e \in \mathcal{P}$;*
- *$p \in \mathcal{P}$ for all $p \in e$, $p > p_e$, such that $l(p_e, p) = 2\delta \cdot k$, for some $k \in \mathbb{N}$;*

- *If* $\exists p' \in \mathcal{P}' \cap e$ *such that* $d(p_e, p') > 2\delta\lfloor l(p_e, v_b)/(2\delta)\rfloor$ *then* $v_b \in \mathcal{P}$.

Moreover, $\lfloor l_e/(2\delta)\rfloor \leq |\mathcal{P} \cap e| \leq \lfloor l_e/(2\delta)\rfloor + 2$. *In particular,*

(i) If $0 \leq q_e \leq \hat{l}_e$, *then* $|\mathcal{P} \cap e| = \lfloor l_e/(2\delta)\rfloor + 1$ *if* $v_b \notin \mathcal{P}$, $|\mathcal{P} \cap e| = \lfloor l_e/(2\delta)\rfloor + 2$ *otherwise.*

(ii) If $\hat{l}_e < q_e \leq 2\delta$, *then* $|\mathcal{P} \cap e| = \lfloor l_e/(2\delta)\rfloor$ *if* $v_b \notin \mathcal{P}$, $|\mathcal{P} \cap e| = \lfloor l_e/(2\delta)\rfloor + 1$ *otherwise.*

Proof. It is easy to observe that $|\mathcal{P}| \leq |\mathcal{P}'|$. First, since facilities are placed every 2δ distance on e, the original covering $\mathcal{P}'$ cannot contain fewer facilities than $\mathcal{P}$. Since the rest of the facilities are just taken from $\mathcal{P}'$, $|\mathcal{P}| \leq |\mathcal{P}'|$. On the other hand, $\mathcal{P}$ has to δ-cover N for the same reason. That is, the facilities of $\mathcal{P}$ that were not in $\mathcal{P}'$ cover at least as much as the ones originally in $\mathcal{P}'$. The last part of the proposition easily follows from construction, and is illustrated by Fig. 9.6. □

Remark 9.22. *The following example shows that the upper bound is tight, that is, there is a case that satisfies* $|\mathcal{P} \cap e| = \lfloor l_e/(2\delta)\rfloor + 2$. *Consider a network with a single edge* e, *and let* $\delta = 1$ *and* $l_e = 3.5$. *Therefore, to cover* e, *we need 3 facilities* $(|\mathcal{P} \cap e| = 3)$. *Because* $\lfloor l_e/(2\delta)\rfloor + 2 = 3$, *the upper bound is tight. Similarly, we can give a tight example for the lower bound, by simply setting* $l_e = 2$ *in the previous network.*

In the following, we present our reduced formulation, which is an adaptation of MILP (9.6). We treat long edges specifically, improving the scalability of our approach. Edges $e \in E$ such that $\delta < l_e \leq 2\delta$ are subdivided into two sub edges of length smaller than δ. Therefore, we assume that, for every $e \in E$, either $l_e \leq \delta$ or $l_e > 2\delta$. In the former case, all the constraints and variables of the model remain unchanged. In the latter, we introduce new variables and constraints to the model, while dropping some of the constraints originally in (9.6). The objective function also needs adaptation. We introduce all these modifications next.

Let $e = (v_a, v_b) \in E$ be a long edge and, for any feasible solution, let q_e be as in Prop. 9.21. That is, when e is a long edge, we use the former variable q_e of MILP (9.6) to represent the position of the left-most facility on e with respect to v_a. The placement variables y_{v_a}, y_{v_b}, and y_e will be used as well, with slightly different meanings to those in (9.6), as we will explain later on. We introduce an indicator variable $u_e \in \{0, 1\}$ to distinguish between two possible ranges in the domain of q_e. If $0 \leq q_e \leq \hat{l}_e$, then $u_e = 0$; otherwise $\hat{l}_e \leq q_e \leq 2\delta$ and $u_e = 1$. This can be modeled with the following constraints:

$$\begin{aligned} q_e &\leq \hat{l}_e(1 - u_e) + 2\delta u_e, \\ q_e &\geq \hat{l}_e u_e. \end{aligned} \tag{9.11}$$

From Prop. 9.21, there is a transition in the number of facilities on e when u_e changes from 0 to 1. Let us denote by $L \subseteq E$ the set of long edges of the network. The objective function of the reduced MILP reads as follows

$$\sum_{f \in \mathcal{F} \setminus L} y_f + \sum_{e \in L} \Big(\left\lceil \frac{l_e}{2\delta} \right\rceil - u_e \Big). \tag{9.12}$$

Note that the coefficients on the last term in the objective already account for the facilities installed at v_a for each $e = (v_a, v_b) \in L$, while they do not do so for v_b. This will condition the values of the placement variables in an optimal solution, namely, $y_{v_a} = 0$ for all $e = (v_a, v_b) \in L$. The facilities installed at these nodes will be tracked by the variables q_e, namely, if $q_e = 0$ then

a facility would be installed at v_a. To complete the modeling of the CSCP, we need to ensure that the covering of the long edge fits into the covering of the rest of the network. Namely, some parts of the network might be covered by facilities placed on e, and part of e (namely its tail or the portion between v_a and p_e of Prop. 9.21) could be covered by facilities placed outside e.

We focus first on the case $0 \leq q_e \leq \hat{l}_e$. We need to ensure that both the segment (v_a, p_e) and the tail of e are covered. For the tail, we know that there is a facility at a distance $\hat{l}_e - q_e$ from v_b. This facility covers a length δ of the remaining fragment on e on its right-hand side, which has a length equal to $\hat{l}_e - q_e$. The rest of such fragment should be covered, which can be imposed by the following constraint:

$$r_{v_b} \geq \hat{l}_e - q_e - \delta \iff r_{v_b} + q_e + \delta \geq \hat{l}_e \qquad \forall e \in L \text{ s.t. } u_e = 0. \tag{9.13}$$

To ensure the covering of the segment (v_a, p_e), we have:

$$r_{v_a} + \delta \geq q_e \qquad \forall e \in L \text{ s.t. } u_e = 0.$$

Let us consider now the case $\hat{l}_e < q_e \leq 2\delta$. There is a facility installed at a distance of $2\delta - q_e + \hat{l}_e$ from v_b. Then, to ensure that the tail of e is covered, we need to cover the fragment between this facility and v_b. Since the facility already covers a length δ on this fragment, the following constraint enforces the covering of the tail:

$$r_{v_b} \geq 2\delta - q_e + \hat{l}_e - \delta \iff r_{v_b} + q_e - \delta \geq \hat{l}_e \qquad \forall e \in L \text{ s.t. } u_e = 1. \tag{9.14}$$

To ensure the covering of the segment (v_a, p_e), we have the same equation as before:

$$r_{v_a} + \delta \geq q_e \qquad \forall e \in L \text{ s.t. } u_e = 1.$$

In summary, the following constraints ensure that the edge e is fully covered:

$$r_{v_a} + \delta \geq q_e \quad \forall e \in L, \tag{9.15}$$

$$r_{v_b} + q_e - (2u_e - 1)\delta \geq \hat{l}_e \quad \forall e \in L. \tag{9.16}$$

Constraint (9.16) gathers (9.13) and (9.14) in a single constraint. The reduced MILP model is as follows (we avoid extended writing of the model for the sake of conciseness):

$$\begin{array}{lcl}
\min & (9.12) & \\
\text{s.t.} & (9.6b), (9.6c), (9.6h) & e \notin L \\
& (9.6f), (9.6g), (9.6k) & e' \notin L \\
& (9.6d), (9.6e), (9.6i), (9.6j), (9.6l), (9.6m), (9.6n), (9.6o), (9.6p), (9.6q) & \\
& (9.11), (9.15), (9.16) & \\
& y_e = 1 & e \in L \quad (9.17) \\
& w_e = 0 & e \in L \quad (9.18)
\end{array}$$

where, if $e \in L$, the term $\tau_{vei}(q_e)$ in (9.6n) is replaced by $d(v, v_i) + \mathbf{1}_{i=a} q_e + \mathbf{1}_{i=b}(2\delta u_e + \hat{l}_e - q_e)$. We enforce (9.17) because e always contains a facility if $e \in L$. On the other hand, we need

to include constraints (9.18) to guarantee that the variables r_{v_a} and r_{v_b} can take positive values. Indeed, if $w_e = 1$ for $e \in L$, it may happen that $x_{v_a} = 1$ or $x_{v_b} = 1$, which will will imply, respectively, $r_{v_a} = 0$ or $r_{v_b} = 0$ due to (9.6l). We compute the complete and partial cover sets in the same way as for the original MILP model. Note that no edge or node can completely cover e if $e \in L$.

The following theorem is on the scalability of the reduced MILP above.

Theorem 9.23. *Given a network $N = (V, E, l)$, the maximum number of variables and constraints of the reduced MILP model only depends on V and E.*

Proof. The number of constraints and variables of the reduced model does not grow with the edge lengths, except for a constant factor of 2 for those edges $e \in E$ such that $\delta < l_e \leq 2\delta$. □

9.7 Computational results

In this section, we present the computational experiments testing the existing and proposed formulations and strengthening techniques for CSCP and its discrete variant (facilities on nodes).

9.7.1 Experiment Setup

We describe the setup of the experiments including the benchmarks, development environment, implementation of algorithms and solution statistics. The computational results and source code are publicly released on our project website: https://github.com/lidingxu/cflg/, where we provide a bash file to reproduce the experiments in Linux systems. Those benchmarks that we generated for this study, or that were publicly available already, are also available at the repository.

Benchmarks. We use three different benchmarking sets: two come from the literature, and the other has been generated synthetically. For every instance, we set the coverage radius δ equal to the average of the edge lengths. We describe these benchmarks next.

`Kgroup`. It consists of 23 prize-collecting Steiner tree problem instances from [208], and the benchmark includes the graphs and edge lengths of these instances. These random geometric instances are designed to have a local structure somewhat similar to street maps. Nodes correspond to random points in the unit square. The number of nodes ranges from 22 to 241. There is an edge between two nodes if their distance is no more than a prescribed threshold which depends on the number of nodes, and the length of an edge is the Euclidean distance between the two points. It is divided into two sets, `Kgroup_A` and `Kgroup_B`. The first one consists of 12 small instances with up to 45 nodes, and the second one consists of 11 large instances with up to 241 nodes.

`City`. It consists of real data of 9 street networks for some German cities, and it was first used in [56]. The number of nodes ranges from 132 to 771. The length of each edge is the length of the underlying street segment.

`Random`. It consists of 24 random network instances generated via Erdős-Rényi binomial method with the package “Networkx" (see [170]). A network is constructed by connecting nodes randomly. Each edge is included with a predefined uniform probability p. The number of nodes, n, is in $\{10, 15, 20, 25, 30, 40\}$. For each n, we generate random graphs with different adjacency probabilities, namely $p \in \{0.1, 0.2, 0.3, 0.4\}$. Furthermore, we split these instances into

two benchmarks: `Random_A` and `Random_B`. `Random_A` contains instances with $n \in \{10, 15, 20\}$. `Random_B` contains instances with $n \in \{25, 30.40\}$.

Coverage radii. For each network, we define two sets of coverage radii: "Small" equal to [Average Edge Length], and "Large" equal to ×2 [Average Edge Length], respectively.

Problem preprocessing. Networks of instances are modified in a problem preprocessing step to be amenable to MILP models.

Given an original network of each instance, in the first preprocessing step, we delete any degree-two node and concatenate its adjacent edges to a new edge, as long as the deletion does not yield a self-loop. Such a node can be treated as an interior point of the new edge. We refer to the preprocessed network without any such degree-two node as the degree-two-free network.

Even after the first preprocessing step, the degree-two-free network may not correspond to the actual problem network to solve, since we may subdivide the degree-two-free network for the non-reduced model to guarantee that $\delta > \max_{e \in E} |e|$. We refer to the preprocessed network after the second preprocessing step as the subdivided network, which is degree-two-free and satisfies $\delta > \max_{e \in E} |e|$. Therefore, the size (number of nodes and edges) of a subdivided network depends on δ.

Development environment. The experiments are conducted on a computer with Intel Core i7-6700K CPU @ 4.00GHZ and 16GB main memory. JuMP [131] is a modeling language for mathematical optimization embedded in Julia. We use JuMP to implement our models and interact with MILP solvers. Specifically, we use ILOG CPLEX 20.1 to solve our models. Alternatively, the implementation allows users to switch easily to other solvers (e.g. Gurobi and GLPK).

CPLEX's parameters are set as their defaults, except that we disable its parallelism and set the MIP absolute gap to 1 (due to the integral objective). The experiments are partitioned into jobs. Every job calls CPLEX to solve an instance, and this job is handled by one process of the multi-core CPU. To safeguard against a potential mutual slowdown of parallel processes, we run only one job per core at a time, and we use at most three processes in parallelism. The time limit of each job is set to 1800 CPU seconds.

Model implementation. We implement six models based on different combinations of formulations and settings. The first five models address CSCP, while the last model solves its discrete restriction, i.e. the variant in which facilities must be placed at nodes. These models are as follows.

`EF`. This model implements the model from [142] for CSCP. This formulation only uses edges to model facility locations, and the authors do not consider the complete and partial cover sets to delimit the size of the model. This model assumes $\delta > \max_{e \in E} |e|$, and it reads the subdivided graph.

`F0`. This model implements a basic formulation that is a simplification of the model (9.6). It does not use the complete and partial cover information nor any of the strengthening techniques in Sect. 9.4. Hence, it does not call the network processing algorithm nodeCover. This model assumes $\delta > \max_{e \in E} |e|$, and it reads the subdivided graph. More precisely, the constraints (9.6b)-(9.6e) related to complete covers are removed, the complete cover variables

Model	Problem	Delimitation	Strengthening	Long edge	Size	Input network	Comment
EF	CSCP	No	No	No	Very large	Subdivided network	From [142]
F0	CSCP	No	No	No	Large	Subdivided network	The simple model
F	CSCP	Yes	No	No	Medium	Subdivided network	The complete model
SF	CSCP	Yes	Yes	No	Medium	Subdivided network	The strengthened model
RF	CSCP	Yes	Yes	Yes	Small	Degree-two-free network	The reduced model
SFD	discrete facility SCP	Yes	Yes	No	Very Small	Subdivided network	The discrete model

Table 9.2 Model summary

w are fixed to 0; for each $v \in V$, the partial cover sets $\mathcal{E}_{\rm p}(v)$, $\mathcal{EI}_{\rm p}(v)$ are solely set, respectively, as E and $E\times\{a,b\}$, and consequently, $M_v = \delta$, $M_{vv'} = r(N)$ for $v' \in \mathcal{E}_{\rm p}(v)$, $M_{ve'i'} = r(N)+|e'|$ for $(e',i') \in \mathcal{EI}_{\rm p}(v)$ are trivial valid bound constants, where $r(N) := \max_{v,v'\in N} d(v,v')$ is the radius of the problem network N.

F. This model implements the complete formulation (9.6) for CSCP, it does use the complete and partial cover information, and hence it calls the network processing algorithm nodeCover. It does not use the strengthening techniques in Sect. 9.4. This model assumes $\delta > \max_{e\in E}|e|$, and it reads the subdivided network as well. For each $v \in V$, due to the delimited partial cover set, $M_v = \delta$, $M_{vv'} = \delta$ for $v' \in \mathcal{E}_{\rm p}(v)$, $M_{ve'i'} = \delta+|e'|$ for $(e',i') \in \mathcal{EI}_{\rm p}(v)$ are valid bound constants.

SF. This model strengthens F by using the techniques described in Sect. 9.4. More precisely, the big-M constants are reduced as Sect. 9.4.1; the "Leafs" inequalities are used to fix variables; and the "Neighborhood" inequalities are implemented as model constraints which replace (9.6f).

RF. This model implements the reduced formulation from Sect. 9.6. It only requires $\delta < 2\max_{e\in E}|e|$. Given a degree-two-free network, it models the long edge specifically as the description Sect. 9.6, and it subdivides the edges with lengths greater than δ and smaller than 2δ into two sub-edges.

SFD. Any solution of the discrete restriction of CSCP—where facilities can only be placed at nodes— is a feasible solution of CSCP. We name this discrete restriction by the discrete facility SCP. This model solves the discrete facility SCP, which solely sets $y_e = 0$ for all $e \in E$ in SF model.

The above models are summarized in Table 9.2. Both EF and F0 consider that any two points in the network can possibly cover each other, and do not utilize the complete and partial cover information. They have been already compared in Sect. 9.3.1, and hence F0 should have fewer variables and constraints than EF. We are interested in the dual gaps obtained after the models are solved within the time limit for these models.

Performance metrics and statistical tests. We describe the performance metrics and the ways to compute their statistics. These statistics will be used to evaluate the model performance.

Let $\underline{v}$ be a dual lower bound and $\overline{v}$ be a primal upper bound obtained after solving some of the models described above, the relative dual gap is defined as:

$$\sigma := \frac{\overline{v}-\underline{v}}{\overline{v}}.$$

A smaller relative dual gap indicates better primal and dual behavior of the model.

Let n_{sd} be the number of nodes of the subdivided network of that instance, note that a trivial primal solution is the set of the nodes of the subdivided network (for which edge length is at most δ). Therefore, to normalize the primal solution value, we define the relative primal

bound

$$v_r := \frac{\overline{v}}{n_{sd}}.$$

If $v_r < 1$, then the model finds a solution better than the trivial one.

In order to evaluate model performance, we compute shifted geometric means (SGMs) of performance metrics, which provides a measure for relative differences. This avoids statistics from being dominated by outliers with large absolute values as is the case for the arithmetic mean. The SGM also avoids an over-representation of results with small absolute values. The SGM of values $v_1, ..., v_M \geq 0$ with shift $s \geq 0$ is defined as

$$\left(\prod_{i=1}^{M}(v_i + s)\right)^{1/M} - s.$$

We say an instance is affected by a model, if solving this model finds a feasible solution; the instance is solved by this model, if solving this model finds an optimal solution. If an instance is unaffected, usually the model is too large to be read into the MILP solver.

We record the following performance metrics of each instance for each model, and compute the benchmark-wise SGMs:

1. t: the total running time in CPU seconds, with a shifted value set to 1 second;
2. σ: the relative dual gap, with a shifted value set to 0.01;
3. v_r: the relative primal bound, with a shifted value set to 0.01.

For an unaffected instance, we set by default $t = 1800$, $\sigma = 1$ and $v_r = 1$. Note that the time does not include the preprocessing time, since we find that the preprocessing is usually at most 0.5 seconds.

We will discuss the computational results, which are divided into two parts. In the first part, we compare the five models `EF`, `F0`, `F`, `SF`, and `RF`. We evaluate the performance metrics of these models. The second part compares `RF` and `SFD`, quantifying the facilities that are saved by allowing continuous location. In the following, we will analyze the aggregated results.

9.7.2 Comparative Analysis of Continuous Models

We compare five continuous models for the CSCP, namely `EF`, `F0`, `F`, `SF`, and `RF`. For each benchmark, radius and model, we record a triple of integers S/A/T: S denotes the number of solved instances, A denotes the number of affected instances, and T denotes the number of total instances in the benchmark. Moreover, we also report the average SGMs of the dual gaps, solving times and relative primal bounds among all instances in the benchmark. Table 9.3 summarizes these results.

First, we notice that `EF` cannot affect any instance in any benchmark; `RF`, `SF` and `RF` can affect all instances, i.e., solutions are provided by these models; `RF` is the model that solves the most number of instances (11), and `SF` is the second best one (10).

Secondly, we compare `EF` and `F0`. `F0` is obviously superior to `EF`. With `F0`, 39 among 56 instances of small radius (resp. 42 among 56 instances of large radius) can be read by the CPLEX solver, while the instances modeled by `EF` are too large to read. Therefore, better

solutions than trivial solutions are found by F0: on average, for instance of small radius (resp. large radius), F0 finds solutions that use 25.2% (resp. 74.1%) fewer facilities than the trivial solution.

Then, we compare F0, F and SF. With the delimitation of complete and partial covering sets, F and SF can affect all instances (especially those in Kgroup_B, of which F0 could just read one). With the strengthening technique, SF has only marginal improvement in the relative primal bound, and solving time, while F is even slightly better than SF in the dual gap. We observe, in our experiment, that adding valid inequalities might slow down the internal solving process of CPLEX.

Finally, we compare SF and RF. RF outperforms SF in all performance metrics. Moreover, RF is the best one among those models affecting all instances. Indeed, for many instances, their degree-two-free networks may contain long edges, and RF avoids introducing too many variables and constraints for modeling their coverage.

Kgroup_B is the hardest benchmark. The best model RF still has an average dual gap of 59.1% and 154.2% relative primal bound for instances of small radius, and this means that RF cannot produce better solutions than the trivial one.

We find that for all the models (except for EF), the average dual gaps and solving times of instances of large radius are smaller than those of instances of small radius. This shows that the large radius has a positive effect on the model performance, and an instance of a small radius may be more difficult than the same instance of a large radius. This is because, with a larger radius, the network after processing is smaller.

In Fig. 9.7, we show scatter plots of the relative dual gaps and the relative primal bounds of affected instances between different settings. For every plot, there is a line in which the points have equal (X,Y)-values. If points fall below the line, then the Y-axis model performs better for the corresponding instances. Note that when comparing F0 and F, the plots do not consider the unaffected instances of F0 which are affected or solved by F. Moreover, F0 even closes more duality gaps than F, but F can find better primal solutions. These plots give an overview of all affected instances and support the above analysis.

To summarize, we have shown that the two proposed techniques— that to delimit the coverage areas from a given point in Sect. 9.2, and that to cover long edges in Sect. 9.6,— can reduce the model size drastically. Among the five models tested, RF features the best overall performance, which is achieved by directly modeling covers on long edges. On the other hand, delimiting the covering sets to the potential, complete and partial covers also reduce the model size, which allows F to read all the tested instances.

9.7.3 Comparative Analysis of Continuous and Discrete Models

In CSCP, the facilities are located either at nodes or edges, while in the discrete variant considered in this section facilities can only be located at nodes. Our objective is to evaluate the number of facilities that can be saved by allowing continuous location. Since the discrete model studied here, SFD, is a discrete restriction of CSCP, every optimal solution is a feasible solution of CSCP. We solve SFD for the discrete facility SCP and compare the results with the best model for CSCP, RF.

Benchmark	Radius	EF				FO			
		time	$\sigma(\%)$	$v_r(\%)$	S/A/T	time	$\sigma(\%)$	$v_r(\%)$	S/A/T
city	Small	1800.0	100.0%	100.0%	0/0/9	1801.7	56.8%	83.3%	0/3/9
	Large	1800.0	100.0%	100.0%	0/0/9	1800.9	42.3%	36.2%	0/6/9
Kgroup_A	Small	1800.0	100.0%	100.0%	0/0/11	1802.6	25.1%	85.0%	0/11/11
	Large	1800.0	100.0%	100.0%	0/0/11	139.2	14.7%	19.2%	7/11/11
Kgroup_B	Small	1800.0	100.0%	100.0%	0/0/12	1800.4	92.6%	98.8%	0/1/12
	Large	1800.0	100.0%	100.0%	0/0/12	1800.1	93.2%	86.6%	0/1/12
random_A	Small	1800.0	100.0%	100.0%	0/0/12	16.8	15.9%	54.8%	9/12/12
	Large	1800.0	100.0%	100.0%	0/0/12	0.2	25.5%	19.5%	12/12/12
random_B	Small	1800.0	100.0%	100.0%	0/0/12	1317.6	36.4%	63.3%	1/12/12
	Large	1800.0	100.0%	100.0%	0/0/12	154.4	26.0%	10.0%	11/12/12
all	Small	1800.0	100.0%	100.0%	0/0/56	625.8	37.4%	74.8%	10/39/56
	Large	1800.0	100.0%	100.0%	0/0/56	132.5	33.1%	25.9%	30/42/56

Benchmark	Radius	F				SF			
		time	$\sigma(\%)$	$v_r(\%)$	S/A/T	time	$\sigma(\%)$	$v_r(\%)$	S/A/T
city	Small	1802.9	29.5%	62.2%	0/9/9	1801.3	30.1%	66.9%	0/9/9
	Large	1801.2	28.4%	21.7%	0/9/9	1800.9	29.1%	21.7%	0/9/9
Kgroup_A	Small	1803.0	33.1%	82.2%	0/11/11	1801.3	32.0%	80.6%	0/11/11
	Large	238.0	18.9%	19.1%	8/11/11	300.8	19.0%	19.1%	8/11/11
Kgroup_B	Small	1800.6	80.8%	240.5%	0/12/12	1801.4	79.7%	191.9%	0/12/12
	Large	1800.4	85.1%	80.5%	0/12/12	1800.7	85.9%	77.3%	0/12/12
random_A	Small	20.2	16.5%	54.3%	9/12/12	16.1	17.1%	54.9%	9/12/12
	Large	0.3	25.5%	19.5%	12/12/12	0.2	10.4%	17.9%	12/12/12
random_B	Small	1574.2	38.8%	64.9%	1/12/12	1501.2	40.0%	67.5%	1/12/12
	Large	220.5	19.9%	10.3%	9/12/12	175.7	18.8%	10.0%	11/12/12
all	Small	675.0	35.2%	86.2%	10/56/56	637.6	35.5%	83.6%	10/56/56
	Large	163.0	30.2%	23.6%	29/56/56	160.9	24.9%	22.8%	31/56/56

Benchmark	Radius	RF			
		time	$\sigma(\%)$	$v_r(\%)$	S/A/T
city	Small	1804.4	16.2%	54.1%	0/9/9
	Large	1801.5	25.8%	21.3%	0/9/9
Kgroup_A	Small	1622.6	21.5%	77.5%	1/11/11
	Large	158.9	19.2%	19.3%	8/11/11
Kgroup_B	Small	1800.9	59.1%	154.2%	0/12/12
	Large	1800.6	75.5%	63.3%	0/12/12
random_A	Small	15.9	8.1%	54.3%	9/12/12
	Large	0.3	26.6%	19.8%	12/12/12
random_B	Small	1304.3	38.5%	63.8%	1/12/12
	Large	190.2	19.8%	11.2%	9/12/12
all	Small	604.9	23.7%	75.4%	11/56/56
	Large	146.6	29.2%	22.8%	29/56/56

Table 9.3 Results for continuous models

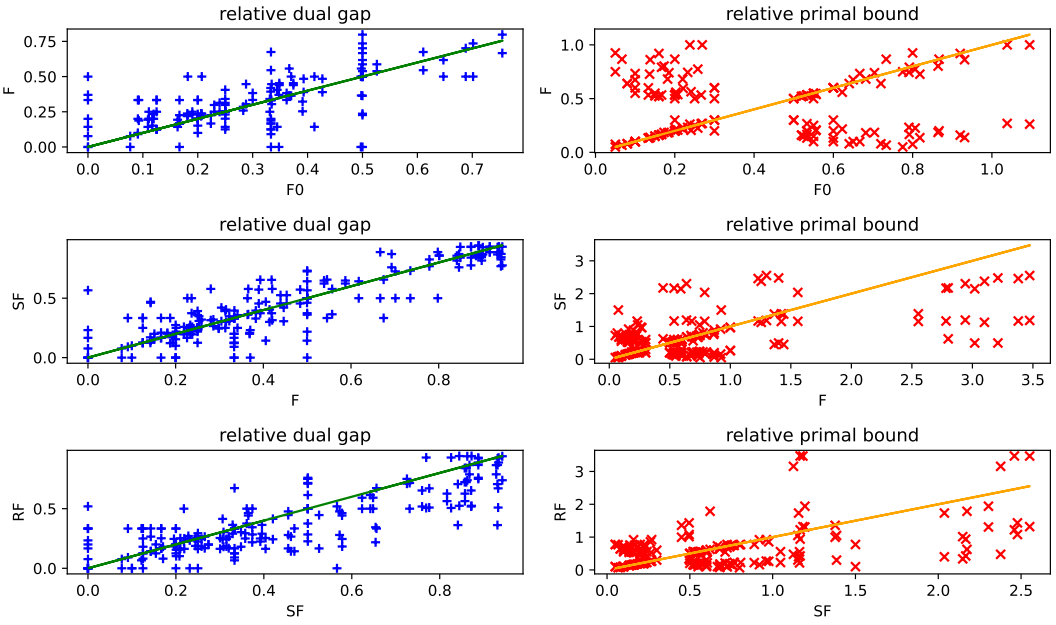


Figure 9.7 Scatter plots of the relative dual gaps and the relative primal bounds between different settings

In addition to the previous performance statistics, we also record for each instance, a new relative primal bound for the continuous model defined as:

$$v'_r := \frac{\overline{v}}{\overline{v_d}},$$

where $\overline{v_d}$ is the best solution found by SFD. If $v'_r < 1$, then the continuous model (in this case, RF) finds a solution better than the one found by the discrete model.

Benchmark	Radius	RF					SFD				
		time	$\sigma(\%)$	$v_r(\%)$	$v'_r(\%)$	S/A/T	time	$\sigma(\%)$	$v_r(\%)$	$v'_r(\%)$	S/A/T
city	Small	1804.4	16.2%	54.1%	89.3%	0/9/9	0.2	0.3%	60.6%	100.0%	9/9/9
	Large	1801.5	25.8%	21.3%	92.0%	0/9/9	3.4	1.1%	23.2%	100.0%	9/9/9
Kgroup_A	Small	1622.6	21.5%	77.5%	91.1%	1/11/11	0.5	2.7%	85.1%	100.0%	11/11/11
	Large	158.9	19.2%	19.3%	85.5%	8/11/11	0.4	6.7%	22.6%	100.0%	11/11/11
Kgroup_B	Small	1800.9	59.1%	154.2%	185.0%	0/12/12	66.1	0.8%	83.3%	100.0%	10/12/12
	Large	1800.6	75.5%	63.3%	312.1%	0/12/12	136.5	1.1%	20.2%	100.0%	12/12/12
random_A	Small	15.9	8.1%	54.3%	86.0%	9/12/12	0.0	1.2%	63.2%	100.0%	12/12/12
	Large	0.3	26.6%	19.8%	93.6%	12/12/12	0.0	2.4%	21.1%	100.0%	12/12/12
random_B	Small	1304.3	38.5%	63.8%	91.8%	1/12/12	1.0	2.1%	69.5%	100.0%	12/12/12
	Large	190.2	19.8%	11.2%	103.3%	9/12/12	1.7	8.7%	10.9%	100.0%	12/12/12
all	Small	604.9	23.7%	75.4%	104.6%	11/56/56	2.2	1.3%	72.1%	100.0%	54/56/56
	Large	146.6	29.2%	22.8%	121.3%	29/56/56	3.9	3.2%	18.7%	100.0%	56/56/56

Table 9.4 Results for continuous and discrete models

Table 9.4 depicts some comparative results. A first observation is that SFD has fewer variables and constraints than RF, as it models a simpler problem. In addition, our strengthening techniques explain that SFD can solve almost all instances in a very short time. Moreover, even the average relative primal bound of SFD is smaller than RF. However, we note that, with

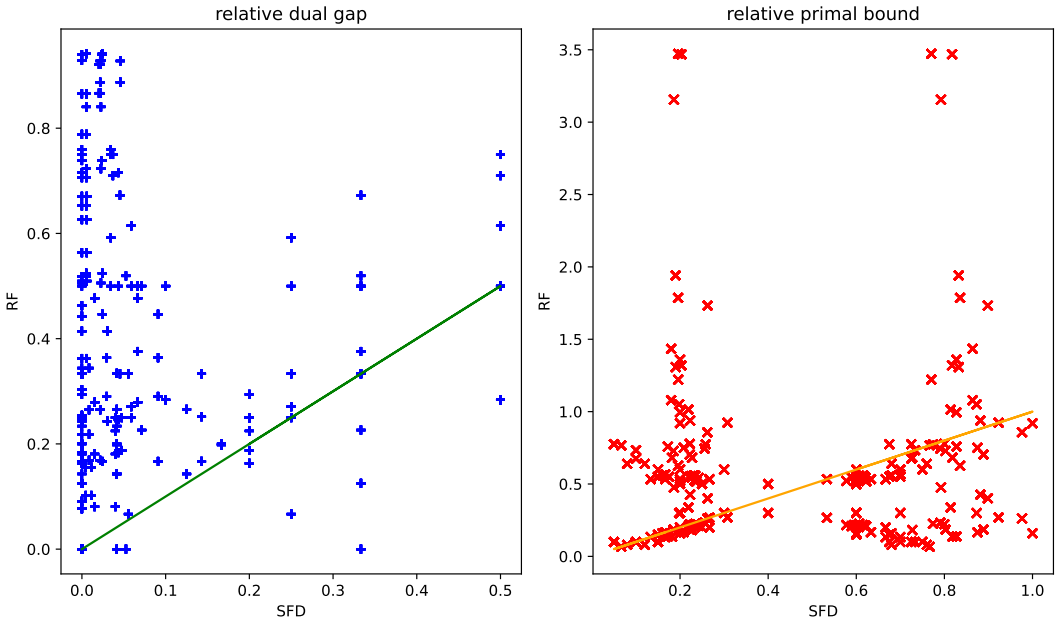


Figure 9.8 Scatter plots of the relative dual gaps and the relative primal bounds between **SFD** and **RF**

the exception of `Kgroup_B` and `random_B` of large radius, `RF` finds solutions with fewer facilities. For `Kgroup_B`, `RF` has a larger average dual gap than `SFD`.

In Fig. 9.8, we also show scatter plots of the relative dual gaps (σ) and the relative primal bounds (v_r) for those instances affected by both `SFD` and `RF`. These plots complement the averaged results of Table 9.4 by giving information on all affected instances, and support the above analysis.

By allowing location at edges, the continuous model can reduce the number of installed facilities. However, it becomes more challenging to solve the problem. The results suggest that calling `SFD` and passing its solution as a warm-start to `RF` can make sense as a two-step optimization approach.

9.8 Conclusions

In this work, we use an integer programming approach to solve CSCP, propose various MILP formulations for this problem and test these formulations against an existing MILP formulation on several benchmarks from the literature.

The existing works mainly consider discretization methods and FDS. Discretization methods are indeed preprocessing procedures that restrict CSCP to an equivalent set-covering problem with continuous demands and candidate facilities in FDS. But FDS is only computable when CSCP satisfies some assumption, so it is not practical to employ FDS for general CSCP. On the other hand, to delimit the search space of MILP models, we use alternative preprocessing procedures. We explore the delimitation that relaxes the concept of FDS, which also restricts the candidate space of facilities (from the full network to a still continuous sub-network). We learn the following ideas to tackle similar problems (possibly with more complex constraints):

i). Integer programming methods are more viable and flexible for general graphs, as a partial delimitation of the problems is useful for strengthening the models (via separation of valid inequalities, tightening big-M constants, variable fixing), while the discretization methods require a full delimitation and a complete characterization of FDS. ii). Our delimitation is applicable as long as the facility location is continuous.

Specifically, we devise and implement four models for CSCP: `F0`, `F` and `SF`, which belong to the same family of models, and `RF`, which is the reduced one. These models mainly differ in preprocessing procedures applied. We find that MILP solvers cannot read or build the MILP model from [142] for any instances in our test bed, and this is due to the model having a large number of constraints and variables. So we mainly compare our four models. We find the MILP size is the main barrier to scalability. The delimitation of those parts of the network that can be covered from a specific location has been revealed as a very effective technique to reduce the model size. In addition, avoiding breaking long edges in the reduced model also results in better scalability. In conclusion, `RF` is the best model: it can find good solutions with a small dual gap.

Meanwhile, the model `SF` is easily cast into `SFD` for the discrete restriction of CSCP. We find that allowing continuous facilities decreases the number of installed facilities but increases the solving time significantly. We note that `SFD` finds an optimal solution for the discrete facility SCP quickly, which is a primal solution for CSCP. Therefore, `SFD` can be called as a fast MILP-based primal heuristic for CSCP.

As for future studies, devising efficient heuristics to be integrated into MILP solvers can be useful to improve the primal performance of the proposed models. For instance, different relaxations of CSCP can be worth exploring, such as that where demand only happens at nodes (i.e. only nodes are to be covered). Every solution of CSCP would be a solution of such relaxation, and hence the optimal value of the latter is a valid dual lower bound of the optimal value of CSCP. If solving this combinatorial relaxation is efficient and provides a stronger dual lower bound than the LP relaxation of CSCP, we can utilize this result and integrate the combinatorial dual bound into the MILP solver, which leads to a combinatorial branch-and-bound algorithm.

Moreover, we can investigate the potential use of FDS to further delimit the search space and integrate FDS in the preprocessing procedure.

Chapter 10

Conclusions and perspective

This chapter summarizes the key findings of this thesis. Towards the conclusion of this chapter, we delve into the perspectives of this thesis work, discussing its potential for uncovering novel applications across diverse domains and its possible avenues for further development.

10.0.1 Conclusions

Within the scope of this thesis, we focus on the examination of relaxation techniques tailored for challenging MINLP problems. Our approach revolves around a structural exploration of effective relaxation strategies suitable for a variety of problem types. In the introductory section, we introduce a comprehensive list of relaxation methods drawn from existing literature. These encompass a range of approaches, such as relaxations stemming from extended formulations, submodularity-based relaxations, PWL relaxations, and intersection cuts. Additionally, we introduce advanced relaxation techniques that build upon intersection cuts and submodularity. These relaxation tools are then employed to address various real-world applications.

Our computational findings demonstrate that enhancing relaxation methods can lead to accelerated performance in exact MINLP solvers. These enhancements are primarily effective for a specific subset of MINLP problems, given that our relaxations are tailored to the structural characteristics of such problems. Furthermore, our relaxation methods are highly adaptable, with the ability to be activated or deactivated through the structural detection functions integrated into MINLP solvers. This adaptability facilitates the seamless integration of relaxation methods within the sBB algorithmic framework. As a result, the research on MINLP becomes more accessible, as even a modest enhancement in relaxation methods can serve as a foundational component for future studies.

It is worth noting that the techniques we present primarily facilitate exact solutions for MINLP problems, albeit at the expense of increased computation time. Consequently, our relaxation methods find their ideal application in static scenarios where problems are solved offline, with a primary focus on achieving high accuracy. In such settings, these relaxation methods function as core algorithms within MINLP solvers, akin to the role of propagation in constraint programming [5].

It is important to recognize that relaxation methods belong to a broader category of approximation methods. While relaxation methods do not directly yield solutions, approximation methods are geared towards delivering approximated solutions within a reasonable time. In this context, approximation methods are considered front-end framework, as exact MINLP

solvers, commonly called upon by users instead of relaxation methods. As a result, relaxation methods can be viewed as the backends of many approximation techniques.

In recent times, the field of MINLP research has encountered several challenges arising from factors such as the emergence of deep learning techniques, the prevalence of large-scale problems, and the inherent dynamics and uncertainties present in mathematical models. Additionally, a frequently overlooked challenge for MINLP research lies in the legacy aspects of its ecosystem. In the subsequent sections, we outline two primary research directions aimed at tackling these challenges.

10.1 Perspectives in algorithm design

The historical influence of MILP techniques might contribute to the existing legacy within the MINLP ecosystem. However, the applicability of these techniques to MINLP problems is still an active research area. In the subsequent sections, we conduct a comprehensive survey of open problems and associated research perspectives. These problems expand the scope of MINLP within the wider realm of mathematical optimization.

10.1.1 Extended formulations vs projected formulations

The prevalent choice among mathematical solvers is to embrace projected formulations of the underlying relaxations, as opposed to extended formulations. This preference can be attributed to several factors. Firstly, these relaxations, notably linear programming (LP) relaxations, can be progressively enhanced by cutting planes, thus offering a straightforward means of controlling relaxation size. Secondly, it is noteworthy, that while an extended formulation with a polynomial-size may exist, it could correspond to an exponentially sized projected formulation. Nonetheless, the practical utilization of extended formulations necessitates a predetermined design to balance the trade-off involving model size and the strength of the model. An open question or ongoing debate centers around the applicability of extended formulations.

The preference over projected formulations could potentially face constraints stemming from computational resources. Given the prevailing capabilities of contemporary computers, there arises an opportunity to reevaluate the applicability of extended formulations. Recent progress in polynomial programming research has unveiled a range of attractive extended formulations, such as the sparse Lasserre hierarchy and relative entropy relaxations [235]. These novel extended formulations introduce a degree of flexibility in representing polynomial optimization.

An approach for integrating extended formulations into solvers involves the dynamic resolution of extended formulations at specific nodes within the branch-and-bound tree. Furthermore, at a given search node, the extended formulations might be adaptively lifted, albeit at the cost of computational resources. The determination of whether to enhance these formulations is intrinsically linked to optimization theory. A comparable challenge is encountered in the context of facial reduction for semi-definite programming, which can be seen as the endeavor to identify a suitable basis for the sum-of-squares representation of a polynomial.

10.1.2 Modeling power: the lack of convexity and nonlinear function types

In the realm of MINLP, the central challenge is tackling non-convex optimization problems, although the methods deployed often hinge upon convex optimization. Particularly within the MINLP landscape, the foundational algorithmic framework rests upon the utilization of convex relaxations.

Yet, the majority of optimization solvers remain firmly rooted in LP relaxations. Particularly, the MINLP solver Mosek can address conic programming problems. Its reference book [18] lists several essential cones, and the solver is capable of representing a significant portion of convex sets through compositional operations. However, it is noteworthy that the study of these fundamental cones and their optimization attributes largely predates the 2000s.

We aim to remain attuned to emerging application trends, uncovering novel function types that have the potential for significant impact. Pursuing this research direction has the potential to substantially expand the modeling power of solvers, enabling them to tackle a broader range of optimization challenges.

This apparent lack of basic convexity types could be attributed to the inherent lack of elementary functions - the sources of non-convexity - which include polynomial, rational, trigonometric, hyperbolic, and exponential functions, possibly including their inverse counterparts. The limitations of the available convexity forms pose a problem for the modelling power of MINLP solvers.

Most MINLP solvers can effectively handle factorizable functions and essentially work as composite functions. An ongoing research attempt is to extend their capacities to handle the demands of modern deep neural networks, which often involve a larger number of relatively simple layers. Several MINLP based approaches [290, 286, 317] are proposed for improving neural networks. A novel approach [42, 80] addresses the challenges of progressively modeling and optimizing for auto machine learning. This perspective aligns closely with a contemporary principle in modern machine learning: the fusion of modeling and optimization. Thus, this perspective can inspire the feature design of MINLP solvers.

Convex functions are fertile ground for ongoing research and offer numerous avenues for exploration. While conventional nonlinear functions are usually defined over scalar variables, a promising direction is to consider functions defined over cones, such as convex functions defined over positive-definite matrices. An abundance of such functions can be discovered in quantum information, since quantum states can be aptly represented by probability-dense matrices—complex positive-definite matrices with unit trace. It is worth noting, however, that despite the richness of this field, the current focus of research efforts is on efficient convex optimization techniques rather than global optimization, primarily due to the scale of dense matrices.

Our goal is to keep pace with emerging trends in applications and discover new types of features that are likely to have a significant impact. Pursuing this line of research has the potential to significantly expand the modelling power of solvers and enable them to handle a broader range of optimization tasks.

10.1.3 Approximations vs. relaxations

Throughout history, the problem domains studied in MILP and combinatorial optimization have overlapped considerably, which has fostered a continuous exchange of ideas and knowledge. One particularly common and widely used technique is that of relaxations. These relaxations act as mediators within algorithms to achieve a feasible solution.

A variety of approximation algorithms in combinatorial optimization, for example, solve relaxations to derive solutions that guarantee optimality. Many of these algorithms were developed for theoretical studies, so they guarantee worst-case performance, but in practice they may perform poorly. In contrast, in the MILP domain, relaxation methods are often integrated into the branch-and-bound search framework, with the relaxation solutions serving as the starting point for heuristics. Exact MILP solvers generally provide optimal solutions to many practical problems within a reasonable time. It should be noted, however, that in the field of combinatorial optimization there are approximation algorithms or heuristics based on structural techniques that go beyond simple relaxation. This raises an intriguing question: Can these alternative techniques be used in the context of MINLP? For example, although there are many tailored convex optimization algorithms, convex relaxations in MINLP solvers are mainly based on primal-dual interior point solvers. Integrating these tailored algorithms to take over specific tasks in MINLP solving, such as preprocessing, could be considered as a possible research direction.

10.2 Perspectives of solver framework

The inherent legacy of the MINLP ecosystem may also stem from earlier software systems efforts. These systems, while valuable, often present challenges to new software developers and emerging researchers. The complexity associated with these systems hinders adoption of recent advances in software engineering and innovations in programming languages. The use of the C language in mathematical optimization programs can lead experienced developers to create code with memory leaks, resulting in lengthy debugging cycles. The evolution of C++ standards leads to frequent updates and the integration of new language features in each version. However, due to the divergence between classical C and modern C++, mathematical solvers cannot benefit from these features, e.g. for efficient memory management.

In today's programming education, the focus is on functional programming languages other than C. Consequently, the pool of new researchers who can develop tailored algorithms for MINLP may be smaller than in other areas of mathematical optimization. A notable recent development is the SCIP team's decision to make it a fully open source protocol. This transformation raises hopes that the broader community will contribute extensively to its further development. One way to close the current gap is to gradually replace the C components of SCIP with a modern programming language. Similarly, the Linux community has begun using Rust to reimplement the Linux kernel, a strategy we can emulate.

Rust is an ahead-of-time compiled programming language that focuses on performance, type safety, and concurrency. It enforces memory safety— without requiring the use of a garbage collector or reference counting. It borrows ideas from functional programming, including static types, immutability, higher-order functions, and algebraic data types. In addition, Rust has a built-in package management system and testing infrastructure, making it a suitable choice for an open source-driven development landscape. We envision an enriching

exploration of the software framework jointly orchestrated by researchers, users, and the open source community.

References

[1] *SCIP Doxygen Documentation: examples/GMI/src/sepa_gmi.c Source File*, July 2023. [Online; accessed 11. Jul. 2023].

[2] *SCIP Doxygen Documentation: How to add separators*, July 2023. [Online; accessed 6. Jul. 2023].

[3] *SCIP Doxygen Documentation: nlhdlr_quadratic.c Source File*, July 2023. [Online; accessed 14. Jul. 2023].

[4] S. ABDUL-NABI, A. KHALIL, P. MARY, AND J. F. HÉLARD, *Efficient network coding solutions for limiting the effect of packet loss*, Eurasip Journal on Wireless Communications and Networking, 2017 (2017), p. 35.

[5] T. ACHTERBERG, T. BERTHOLD, T. KOCH, AND K. WOLTER, *Constraint integer programming: A new approach to integrate CP and MIP*, in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5015 LNCS, Springer, 2008, pp. 6–20.

[6] W. P. ADAMS AND H. D. SHERALI, *A tight linearization and an algorithm for zero-one quadratic programming problems*, Management Science, 32 (1986), pp. 1274–1290.

[7] B. ADENSO-DÍAZ AND F. RODRÍGUEZ, *A simple search heuristic for the mclp: application to the location of ambulance bases in a rural region*, Omega, 25 (1997), p. 181–187.

[8] S. AHMED AND A. ATAMTÜRK, *Maximizing a class of submodular utility functions*, Mathematical programming, 128 (2011), pp. 149–169.

[9] İ. AKGÜN, F. GÜMÜŞBUĞA, AND B. TANSEL, *Risk based facility location by using fault tree analysis in disaster management*, Omega, 52 (2015), pp. 168–179.

[10] G. A. AKPAKWU, B. J. SILVA, G. P. HANCKE, AND A. M. ABU-MAHFOUZ, *A survey on 5G networks for the internet of things: Communication technologies and challenges*, IEEE Access, 6 (2017), pp. 3619–3647.

[11] F. A. AL-KHAYYAL AND J. E. FALK, *Jointly constrained biconvex programming*, Mathematics of Operations Research, 8 (1983), pp. 273–286.

[12] A. ALLMAN AND Q. ZHANG, *Branch-and-price for a class of nonconvex mixed-integer nonlinear programs*, Journal of Global Optimization, 81 (2021), pp. 861–880.

[13] N. C. ALMEIDA, R. P. ROLLE, E. P. GODOY, P. FERRARI, AND E. SISINNI, *Proposal of a hybrid LoRa mesh / LoRaWAN network*, in 2020 IEEE International Workshop on Metrology for Industry 4.0 IoT, 2020, pp. 702–707.

[14] E. ÁLVAREZ-MIRANDA, V. CACCHIANI, T. DORNETH, M. JÜNGER, F. LIERS, A. LODI, T. PARRIANI, AND D. R. SCHMIDT, *Models and algorithms for robust network design with several traffic scenarios*, in Combinatorial Optimization, A. R. Mahjoub, V. Markakis, I. Milis, and V. T. Paschos, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 261–272.

[15] F. ALVELOS AND J. M. V. DE CARVALHO, *Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem*, in International Network Optimization Conference, sn, 2003, pp. 7–12.

[16] K. ANDERSEN, Q. LOUVEAUX, AND R. WEISMANTEL, *An analysis of mixed integer linear sets based on lattice point free convex sets*, Mathematics of Operations Research, 35 (2010), pp. 233–256.

[17] K. ANDERSEN, Q. LOUVEAUX, R. WEISMANTEL, AND L. A. WOLSEY, *Inequalities from two rows of a simplex tableau*, in Integer Programming and Combinatorial Optimization, M. Fischetti and D. P. Williamson, eds., Berlin, Heidelberg, 2007, Springer Berlin Heidelberg, pp. 1–15.

[18] M. APS, *Mosek modeling cookbook*, 2018.

[19] A. ATAMTÜRK AND A. GÓMEZ, *Submodularity in conic quadratic mixed 0–1 optimization*, Operations Research, 68 (2020), pp. 609–630.

[20] ——, *Supermodularity and valid inequalities for quadratic optimization with indicators*, Mathematical Programming, (2022), pp. 1–44.

[21] A. ATAMTÜRK AND V. NARAYANAN, *Polymatroids and mean-risk minimization in discrete optimization*, Operations Research Letters, 36 (2008), pp. 618–622.

[22] A. ATAMTÜRK AND V. NARAYANAN, *The submodular knapsack polytope*, Discrete Optimization, 6 (2009), pp. 333–344.

[23] A. ATAMTÜRK AND V. NARAYANAN, *Submodular function minimization and polarity*, Mathematical Programming, (2021).

[24] A. ATAMTÜRK AND D. RAJAN, *On splittable and unsplittable flow capacitated network design arc-set polyhedra*, Mathematical Programming, Series B, 92 (2002), pp. 315–333.

[25] E. BACCELLI, J. A. CORDERO, AND P. JACQUET, *Multi-point relaying techniques with ospf on ad hoc networks*, in 2009 Fourth International Conference on Systems and Networks Communications, 2009, pp. 53–62.

[26] ——, *Optimization of critical data synchronization via link overlay rng in mobile ad hoc networks*, in The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010), IEEE, 2010, pp. 402–411.

[27] E. BACCELLI, J. A. C. FUERTES, AND P. JACQUET, *Ospf over multi-hop ad hoc wireless communications*, International Journal of Computer Networks & Communications, 2 (2010), pp. 37–56.

[28] S. A. BAHARUDIN, M. F. ZUHAIRI, AND H. DAO, *Impact of interference on wireless mesh network performance*, ARPN Journal of Engineering and Applied Sciences, 14 (2019), pp. 1389–1395.

[29] E. BALAS, *Disjunctive Programming*, Springer International Publishing, Cham, Switzerland.

[30] ——, *Intersection cuts—a new type of cutting planes for integer programming*, Operations Research, 19 (1971), pp. 19–39.

[31] ——, *Intersection cuts—a new type of cutting planes for integer programming*, Operations Research, 19 (1971), pp. 19–39.

[32] M. BALDOMERO-NARANJO, J. KALCSICS, A. MARÍN, AND A. M. RODRÍGUEZ-CHÍA, *Upgrading edges in the maximal covering location problem*, European Journal of Operational Research, 303 (2022), pp. 14–36.

[33] M. BALDOMERO-NARANJO, J. KALCSICS, AND A. RODRÍGUEZ-CHÍA, *Minmax regret maximal covering location problems with edge demands*, Computers & Operations Research, 130 (2021), p. 105181.

[34] M. BANSAL AND K. KIANFAR, *Planar maximum coverage location problem with partial coverage and rectangular demand and service zones*, INFORMS Journal on Computing, 29 (2017), pp. 152–169.

[35] X. BAO, A. KHAJAVIRAD, N. V. SAHINIDIS, AND M. TAWARMALANI, *Global optimization of nonconvex problems with multilinear intermediates*, Mathematical Programming Computation, 7 (2015), pp. 1–37.

[36] C. BARNHART, C. A. HANE, AND P. H. VANCE, *Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems*, Operations Research, 48 (2000), pp. 318–326.

[37] C. BARNHART, E. L. JOHNSON, G. L. NEMHAUSER, M. W. SAVELSBERGH, AND P. H. VANCE, *Branch-and-price: Column generation for solving huge integer programs*, Operations Research, 46 (1998), pp. 316–329.

[38] A. BASU, M. CONFORTI, G. CORNUÉJOLS, AND G. ZAMBELLI, *Maximal lattice-free convex sets in linear subspaces*, Mathematics of Operations Research, 35 (2010), pp. 704–720.

[39] A. BASU, M. CONFORTI, G. CORNUÉJOLS, AND G. ZAMBELLI, *Maximal lattice-free convex sets in linear subspaces*, Mathematics of Operations Research, 35 (2010), pp. 704–720.

[40] A. BASU, S. S. DEY, AND J. PAAT, *Nonunique lifting of integer variables in minimal inequalities*, SIAM Journal on Discrete Mathematics, 33 (2019), pp. 755–783.

[41] S. BATUN, B. T. DENTON, T. R. HUSCHKA, AND A. J. SCHAEFER, *Operating room pooling and parallel surgery processing under uncertainty*, INFORMS Journal on Computing, 23 (2011), pp. 220–237.

[42] G. BAUDART, M. HIRZEL, K. KATE, P. RAM, A. SHINNAR, AND J. TSAY, *Pipeline combinators for gradual automl*, Advances in Neural Information Processing Systems, 34 (2021), pp. 19705–19718.

[43] P.-O. BAUGUION, C. D'AMBROSIO, AND L. LIBERTI, *Maximum concurrent flow with incomplete data*, in Combinatorial Optimization, J. Lee, G. Rinaldi, and A. R. Mahjoub, eds., Cham, 2018, Springer International Publishing, pp. 77–88.

[44] M. BELAIDOUNI AND W. BEN-AMEUR, *On the minimum cost multiple-source unsplittable flow problem*, RAIRO Oper. Res., 41 (2007), pp. 253–273.

[45] J. BELL, S. GRIFFIS, W. CUNNINGHAM, AND J. EBERLAN, *Location optimization of strategic alert sites for homeland defense*, Omega, 39 (2011), p. 151–158.

[46] P. BELOTTI, J. C. GÓEZ, I. PÓLIK, T. K. RALPHS, AND T. TERLAKY, *A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization*, in Numerical Analysis and Optimization, Springer, 2015, pp. 1–35.

[47] P. BELOTTI, C. KIRCHES, S. LEYFFER, J. LINDEROTH, J. LUEDTKE, AND A. MAHAJAN, *Mixed-integer nonlinear optimization*, Acta Numerica, 22 (2013), pp. 1–131.

[48] P. BELOTTI, J. LEE, L. LIBERTI, F. MARGOT, AND A. WÄCHTER, *Branching and bounds tighteningtechniques for non-convex minlp*, Optimization Methods & Software, 24 (2009), pp. 597–634.

[49] ——, *Branching and bounds tighteningtechniques for non-convex minlp*, Optimization Methods & Software, 24 (2009), pp. 597–634.

[50] W. BEN-AMEUR AND J. NETO, *Acceleration of cutting-plane and column generation algorithms: Applications to network design*, Networks, 49 (2007), pp. 3–17.

[51] A. BEN-TAL AND A. NEMIROVSKI, *On polyhedral approximations of the second-order cone*, Mathematics of Operations Research, 26 (2001), pp. 193–205.

[52] O. BENRHAIEM AND L. CHAARI, *Information transmission based on network coding over wireless networks: A survey*, Telecommunication Systems, 65 (2017), pp. 551–565.

[53] D. BERGMAN, *An exact algorithm for the quadratic multiknapsack problem with an application to event seating*, INFORMS Journal on Computing, 31 (2019), pp. 477–492.

[54] D. BERJÓN, G. GALLEGO, C. CUEVAS, F. MORÁN, AND N. GARCÍA, *Optimal piecewise linear function approximation for gpu-based applications*, IEEE Transactions on Cybernetics, 46 (2015), pp. 2584–2595.

[55] O. BERMAN, Z. DREZNER, AND D. KRASS, *Big segment small segment global optimization algorithm on networks*, Networks, 58 (2011), pp. 1–11.

[56] O. BERMAN, J. KALCSICS, AND D. KRASS, *On covering location problems on networks with edge demand*, Computers & Operations Research, 74 (2016), pp. 214–227.

[57] O. BERMAN, J. KALCSICS, AND D. KRASS, *On covering location problems on networks with edge demand*, Computers & Operations Research, 74 (2016), pp. 214–227.

[58] T. BERTHOLD, *Heuristic algorithms in global MINLP solvers*, Verlag Dr. Hut, 2015.

[59] T. BERTHOLD, S. HEINZ, AND S. VIGERSKE, *Extending a CIP framework to solve MIQCPs*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., New York, NY, 2012, Springer New York, pp. 427–444.

[60] D. BERTSIMAS, V. GUPTA, AND N. KALLUS, *Robust sample average approximation*, Mathematical Programming, 171 (2018), pp. 217–282.

[61] K. BESTUZHEVA, M. BESANÇON, W.-K. CHEN, A. CHMIELA, T. DONKIEWICZ, J. VAN DOORNMALEN, L. EIFLER, O. GAUL, G. GAMRATH, A. GLEIXNER, ET AL., *The SCIP optimization suite 8.0*, arXiv preprint arXiv:2112.08872, (2021).

[62] K. BESTUZHEVA, A. CHMIELA, B. MÜLLER, F. SERRANO, S. VIGERSKE, AND F. WEGSCHEIDER, *Global optimization of mixed-integer nonlinear programs with scip 8*, arXiv preprint arXiv:2301.00587, (2023).

[63] D. BIENSTOCK, C. CHEN, AND G. MUÑOZ, *Intersection cuts for polynomial optimization*, in Integer Programming and Combinatorial Optimization, A. Lodi and V. Nagarajan, eds., Cham, 2019, Springer International Publishing, pp. 72–87.

[64] D. BIENSTOCK, C. CHEN, AND G. MUNOZ, *Outer-product-free sets for polynomial optimization and oracle-based cuts*, Mathematical Programming, 183 (2020), pp. 105–148.

[65] A. BILLIONNET AND M. MINOUX, *Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions*, Discrete Applied Mathematics, 12 (1985), pp. 1–11.

[66] K. M. BJÖRK AND T. WESTERLUND, *Global optimization of heat exchanger network synthesis problems with and without the isothermal mixing assumption*, Computers and Chemical Engineering, 26 (2002), pp. 1581–1593.

[67] V. BLANCO, R. GÁZQUEZ, D. PONCE, AND J. PUERTO, *A branch-and-price approach for the continuous multifacility monotone ordered median problem*, European Journal of Operational Research, 306 (2023), pp. 105–126.

[68] R. BLANQUERO, E. CARRIZOSA, AND G. BOGLÁRKA, *Maximal covering location problems on networks with regional demand*, Omega, 64 (2016), pp. 77–85.

[69] G. E. BLAU AND D. J. WILDE, *A Lagrangian algorithm for equality constrained generalized polynomial optimization*, AIChE Journal, 17 (1971), pp. 235–240.

[70] C. BLIEK, P. BONAMI, AND A. LODI, *Solving mixed-integer quadratic programming problems with IBM-CPLEX: A progress report*, in Proceedings of the twenty-sixth RAMP Symposium, 2014, pp. 16–17.

[71] P. BONAMI AND A. TRAMONTANI, *Recent improvement to MISOCP in CPLEX*, tech. rep., 2015.

[72] M. BOUHTOU, S. GAUBERT, AND G. SAGNOL, *Submodularity and randomized rounding techniques for optimal experimental design*, Electronic Notes in Discrete Mathematics, 36 (2010), pp. 679–686.

[73] D. Briskorn and M. Dienstknecht, *Covering polygons with discs: The problem of crane selection and location on construction sites*, Omega, 97 (2020), p. 102114.

[74] V. Bucarey, B. Fortz, N. González-Blanco, M. Labbé, and J. A. Mesa, *Benders decomposition for network design covering problems*, Computers & Operations Research, 137 (2022), p. 105417.

[75] M. R. Bussieck, A. S. Drud, and A. Meeraus, *Minlplib—a collection of test models for mixed-integer nonlinear programming*, INFORMS Journal on Computing, 15 (2003), pp. 114–119.

[76] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello, *Knapsack problems-an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems*, Computers & Operations Research, (2022), p. 105693.

[77] S. Cafieri, J. Lee, and L. Liberti, *On convex relaxations of quadrilinear terms*, Journal of Global Optimization, 47 (2010), pp. 661–685.

[78] A. Caprara, D. Pisinger, and P. Toth, *Exact solution of the quadratic knapsack problem*, INFORMS Journal on Computing, 11 (1999), pp. 125–137.

[79] B. Cardoen, E. Demeulemeester, and J. Beliën, *Operating room planning and scheduling: A literature review*, European journal of operational research, 201 (2010), pp. 921–932.

[80] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener, *Omlt: Optimization & machine learning toolkit*, The Journal of Machine Learning Research, 23 (2022), pp. 15829–15836.

[81] A. Ceselli, L. Létocart, and E. Traversi, *Dantzig–Wolfe reformulations for binary quadratic problems*, Mathematical Programming Computation, (2022), pp. 1–36.

[82] V. Chandrasekaran and P. Shah, *Relative entropy relaxations for signomial optimization*, SIAM Journal on Optimization, 26 (2016), pp. 1147–1173.

[83] P. Chaporkar, K. Kar, and S. Sarkar, *Throughput guarantees through maximal scheduling in wireless networks*, 43rd Annual Allerton Conference on Communication, Control and Computing 2005, 3 (2005), pp. 1557–1567.

[84] R. Chares, *Cones and interior-point algorithms for structured convex optimization involving powers and exponentials*, PhD thesis, UCL-Université Catholique de Louvain Louvain-la-Neuve, Belgium, 2009.

[85] A. Charnes and W. W. Cooper, *Deterministic equivalents for optimizing and satisficing under chance constraints*, Operations Research, 11 (1963), pp. 18–39.

[86] J.-S. Chen and C.-H. Huang, *A note on convexity of two signomial functions*, Journal of Nonlinear and Convex Analysis, 10 (2009), pp. 429–435.

[87] R. Chen, S. Dash, and O. Günlük, *Multilinear sets with two monomials and cardinality constraints*, Discrete Applied Mathematics, 324 (2023), pp. 67–79.

[88] X. Cheng, Q. Wang, Q. Wang, and D. Wang, *A high-reliability relay algorithm based on network coding in multi-hop wireless networks*, Wireless Networks, 25 (2019), pp. 1557–1566.

[89] A. Chmiela, G. Muñoz, and F. Serrano, *On the implementation and strengthening of intersection cuts for qcqps*, Mathematical Programming, (2022), pp. 1–38.

[90] A. Chmiela, G. Muñoz, and F. Serrano, *Monoidal strengthening and unique lifting in miqcps*, in Integer Programming and Combinatorial Optimization: 24th International Conference, IPCO 2023, 2023. accepted for publication.

[91] R. Church and M. Meadows, *Location modeling utilizing maximum service distance criteria*, Geographical Analysis, 11 (1979), pp. 358–373.

[92] R. CHURCH AND C. REVELLE, *The maximal covering location problem*, Papers of the Regional Science Association, 32 (1979), p. 101–118.

[93] R. L. CHURCH AND Z. DREZNER, *Review of obnoxious facilities location problems*, Computers & Operations Research, 138 (2022), p. 105468.

[94] R. L. CHURCH AND R. S. GARFINKEL, *Locating an obnoxious facility on a network*, Transportation Science, 12 (1978), pp. 107–118.

[95] C. COEY, M. LUBIN, AND J. P. VIELMA, *Outer approximation with conic certificates for mixed-integer convex problems*, Mathematical Programming Computation, 12 (2020), pp. 249–293.

[96] ——, *Outer approximation with conic certificates for mixed-integer convex problems*, Mathematical Programming Computation, 12 (2020), pp. 249–293.

[97] M. C. COHEN, P. W. KELLER, V. MIRROKNI, AND M. ZADIMOGHADDAM, *Overcommitment in cloud services: Bin packing with chance constraints*, Management Science, 65 (2019), pp. 3255–3271.

[98] M. CONFORTI AND G. CORNUÉJOLS, *Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem*, Discrete applied mathematics, 7 (1984), pp. 251–274.

[99] M. CONFORTI, G. CORNUÉJOLS, AND G. ZAMBELLI, *Integer programming*, Springer International Publishing, Cham, 2014.

[100] M. CONFORTI, G. CORNUÉJOLS, G. ZAMBELLI, ET AL., *Integer programming*, vol. 271, Springer, 2014.

[101] M. CONFORTI, G. CORNUÉJOLS, A. DANIILIDIS, C. LEMARÉCHAL, AND J. MALICK, *Cut-Generating Functions and S-Free Sets*, Mathematics of Operations Research, 40 (2015), pp. 276–391.

[102] M. CONFORTI, G. CORNUÉJOLS, AND G. ZAMBELLI, *Corner polyhedron and intersection cuts*, Surveys in Operations Research and Management Science, 16 (2011), pp. 105–120.

[103] S. CONIGLIO, F. D'ANDREAGIOVANNI, AND F. FURINI, *A lexicographic pricer for the fractional bin packing problem*, Operations Research Letters, 47 (2019), pp. 622–628.

[104] S. CONIGLIO, F. FURINI, AND I. LJUBIĆ, *Submodular maximization of concave utility functions composed with a set-union operator with applications to maximal covering location problems*, Mathematical Programming, (2022), pp. 1–48.

[105] J.-F. CORDEAU, F. FURINI, AND I. LJUBIĆ, *Benders decomposition for very large scale partial set covering and maximal covering location problems*, European Journal of Operational Research, 275 (2019), pp. 882–896.

[106] J. A. CORDERO, *Link-state routing optimization for compound autonomous systems in the internet*, PhD thesis, École Polytechnique, 2011.

[107] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to algorithms*, MIT press, 2009.

[108] G. CORNUÉJOLS, F. MARGOT, AND G. NANNICINI, *On the safety of gomory cut generators*, Mathematical Programming Computation, 5 (2013), pp. 345–395.

[109] G. CORNUÉJOLS, L. WOLSEY, AND S. YILDIZ, *Sufficiency of cut-generating functions*, Mathematical Programming, 152 (2015), pp. 643–651.

[110] A. COSTA AND L. LIBERTI, *Relaxations of multilinear convex envelopes: dual is better than primal*, in International Symposium on Experimental Algorithms, Springer, 2012, pp. 87–98.

[111] Y. CRAMA, *Concave extensions for nonlinear 0–1 maximization problems*, Mathematical Programming, 61 (1993), pp. 53–60.

[112] Y. CRAMA AND P. L. HAMMER, *Boolean functions: Theory, algorithms, and applications*, Cambridge University Press, 2011.

[113] C. D'AMBROSIO, J. LEE, AND A. WÄCHTER, *An algorithmic framework for MINLP with separable non-convexity*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., New York, NY, 2012, Springer New York, pp. 315–347.

[114] B. D. DEEBAK AND F. AL-TURJMAN, *A hybrid secure routing and monitoring mechanism in IoT-based wireless sensor networks*, Ad Hoc Networks, 97 (2020), p. 102022.

[115] A. DEL PIA AND A. KHAJAVIRAD, *A polyhedral study of binary polynomial programs*, Mathematics of Operations Research, 42 (2017), pp. 389–410.

[116] ——, *The multilinear polytope for acyclic hypergraphs*, SIAM Journal on Optimization, 28 (2018), pp. 1049–1076.

[117] A. DEL PIA, A. KHAJAVIRAD, AND N. V. SAHINIDIS, *On the impact of running intersection inequalities for globally solving polynomial optimization problems*, Mathematical programming computation, 12 (2020), pp. 165–191.

[118] A. DEL PIA AND M. WALTER, *Simple odd-cycle inequalities for binary polynomial optimization*, in International Conference on Integer Programming and Combinatorial Optimization, Springer, 2022, pp. 181–194.

[119] A. DEL PIA AND R. WEISMANTEL, *Relaxations of mixed integer sets from lattice-free polyhedra*, 4OR, 10 (2012), pp. 221–244.

[120] M. DELORME, M. IORI, AND S. MARTELLO, *Bin packing and cutting stock problems: Mathematical models and exact algorithms*, European Journal of Operational Research, 255 (2016), pp. 1–20.

[121] Y. DENG, S. SHEN, AND B. DENTON, *Chance-constrained surgery planning under conditions of limited and ambiguous data*, INFORMS Journal on Computing, 31 (2019), pp. 559–575.

[122] ——, *Chance-constrained surgery planning under conditions of limited and ambiguous data*, INFORMS Journal on Computing, 31 (2019), pp. 559–575.

[123] B. T. DENTON, A. J. MILLER, H. J. BALASUBRAMANIAN, AND T. R. HUSCHKA, *Optimal allocation of surgery blocks to operating rooms under uncertainty*, Operations Research, 58 (2010), pp. 802–816.

[124] G. DESAULNIERS, J. DESROSIERS, AND M. M. SOLOMON, eds., *Column generation*, Springer US, Boston, MA, 2005.

[125] S. S. DEY AND L. A. WOLSEY, *Lifting integer variables in minimal inequalities corresponding to lattice-free triangles*, in Integer Programming and Combinatorial Optimization, A. Lodi, A. Panconesi, and G. Rinaldi, eds., Berlin, Heidelberg, 2008, Springer Berlin Heidelberg, pp. 463–475.

[126] Y. DINITZ, N. GARG, AND M. X. GOEMANS, *On the single-source unsplittable flow problem*, Annual Symposium on Foundations of Computer Science - Proceedings, 19 (1998), pp. 290–299.

[127] M. DRESSLER AND R. MURRAY, *Algebraic perspectives on signomial optimization*, SIAM Journal on Applied Algebra and Geometry, 6 (2022), pp. 650–684.

[128] T. DREZNER, Z. DREZNER, AND A. SCHÖBEL, *The weber obnoxious facility location model: A big arc small arc approach*, Computers & Operations Research, 98 (2018), pp. 240–250.

[129] Z. DREZNER, P. KALCZYNSKI, AND S. SALHI, *The planar multiple obnoxious facilities location problem: A voronoi based heuristic*, Omega, 87 (2019), pp. 105–116.

[130] R. J. DUFFIN, *Linearizing geometric programs*, SIAM review, 12 (1970), pp. 211–227.

[131] I. DUNNING, J. HUCHETTE, AND M. LUBIN, *Jump: A modeling language for mathematical optimization*, SIAM Review, 59 (2017), pp. 295–320.

[132] J. G. ECKER AND M. KUPFERSCHMID, *An ellipsoid algorithm for nonlinear programming*, Mathematical programming, 27 (1983), pp. 83–106.

[133] J. EDMONDS, *Submodular functions, matroids, and certain polyhedra*, in Combinatorial Optimization—Eureka, You Shrink!, Springer, 2003, pp. 11–26.

[134] EN:USER:CBURNETT, *Hamming distance 3 bit binary*, 2007. The image is licensed under CC BY-SA 3.0.

[135] J. E. FALK AND K. R. HOFFMAN, *A successive underestimation method for concave minimization problems*, Mathematics of Operations Research, 1 (1976), pp. 251–259.

[136] A. A. FARLEY, *Note on bounding a class of linear programming problems, including cutting stock problems*, Operations Research, 38 (1990), pp. 922–923.

[137] M. FISCHETTI, I. LJUBIĆ, M. MONACI, AND M. SINNL, *On the use of intersection cuts for bilevel optimization*, Mathematical Programming, 172 (2018), pp. 77–103.

[138] M. FISCHETTI AND M. MONACI, *A branch-and-cut algorithm for mixed-integer bilinear programming*, European Journal of Operational Research, 282 (2020), pp. 506–514.

[139] R. FORTET, *Applications de l'algebre de boole en recherche opérationelle*, Revue Française de Recherche Opérationelle, 4 (1960), pp. 17–26.

[140] C. FRAGOULI, J. WIDMER, AND J. Y. LE BOUDEC, *Efficient broadcasting using network coding*, IEEE/ACM Transactions on Networking, 16 (2008), pp. 450–463.

[141] A. FRANGIONI AND B. GENDRON, *A stabilized structured dantzig–wolfe decomposition method*, Mathematical Programming, 140 (2013), pp. 45–76.

[142] N. FRÖHLICH, A. MAIER, AND H. HAMACHER, *Covering edges in networks*, Networks, 75 (2020), pp. 278–290.

[143] S. FUJISHIGE, *Submodular functions and optimization*, Elsevier, 2005.

[144] T. FUKUNAGA, *Covering problems in edge-and node-weighted graphs*, Discrete Optimization, 20 (2016), pp. 40–61.

[145] F. FURINI AND E. TRAVERSI, *Theoretical and computational study of several linearisation techniques for binary quadratic problems*, Annals of Operations Research, 279 (2019), pp. 387–411.

[146] K. FYSARAKIS, I. ASKOXYLAKIS, O. SOULTATOS, I. PAPAEFSTATHIOU, C. MANIFAVAS, AND V. KATOS, *Which iot protocol? comparing standardized approaches over a common m2m application*, in 2016 IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–7.

[147] G. GAMRATH, D. ANDERSON, K. BESTUZHEVA, W.-K. CHEN, L. EIFLER, M. GASSE, P. GEMANDER, A. GLEIXNER, L. GOTTWALD, K. HALBIG, G. HENDEL, C. HOJNY, T. KOCH, P. LE BODIC, S. J. MAHER, F. MATTER, M. MILTENBERGER, E. MÜHMER, B. MÜLLER, M. E. PFETSCH, F. SCHLÖSSER, F. SERRANO, Y. SHINANO, C. TAWFIK, S. VIGERSKE, F. WEGSCHEIDER, D. WENINGER, AND J. WITZIG, *The SCIP Optimization Suite 7.0*, Technical Report 05, Optimization Online, 2020.

[148] S. GARCÍA AND A. MARÍN, *Covering location problems*, in Location Science, G. Laporte, S. Nickel, and F. S. da Gama, eds., Springer, 2nd edition, 2019, pp. 99–119.

[149] B. GEISSLER, A. MARTIN, A. MORSI, AND L. SCHEWE, *Using piecewise linear functions for solving MINLPs*, in Mixed Integer Nonlinear Programming, J. Lee and S. Leyffer, eds., New York, NY, 2012, Springer New York, pp. 287–314.

[150] S. GÉLINAS, M. DESROCHERS, J. DESROSIERS, AND M. M. SOLOMON, *A new branching strategy for time constrained routing problems with application to backhauling*, Annals of Operations Research, 61 (1995), pp. 91–109.

[151] B. GENDRON, *Revisiting Lagrangian relaxation for network design*, Discrete Applied Mathematics, 261 (2019), pp. 203–218.

[152] B. GENDRON AND M. LAROSE, *Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design*, EURO Journal on Computational Optimization, 2 (2014), pp. 55–75.

[153] L. E. GHAOUI, M. OKS, AND F. OUSTRY, *Worst-case value-at-risk and robust portfolio optimization: A conic programming approach*, Operations Research, 51 (2003), pp. 543–556.

[154] P. C. GILMORE AND R. E. GOMORY, *A linear programming approach to the cutting-stock problem*, Operations Research, 9 (1961), pp. 849–859.

[155] A. GLEIXNER, S. J. MAHER, B. MÜLLER, AND J. P. PEDROSO, *Price-and-verify: a new algorithm for recursive circle packing using Dantzig–Wolfe decomposition*, Annals of Operations Research, 284 (2020), pp. 527–555.

[156] F. GLOVER, *Convexity cuts and cut search*, Operations Research, 21 (1973), pp. 123–134.

[157] M. X. GOEMANS, S. GUPTA, AND P. JAILLET, *Discrete newton's algorithm for parametric submodular function minimization*, in Integer Programming and Combinatorial Optimization, F. Eisenbrand and J. Koenemann, eds., Cham, 2017, Springer International Publishing, pp. 212–227.

[158] C. GOMEZ, P. SALVATELLA, O. ALONSO, AND J. PARADELLS, *Adapting aodv for ieee 802.15.4 mesh sensor networks: theoretical discussion and performance evaluation in a real environment*, in 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06), 2006, pp. 9 pp.–170.

[159] R. E. GOMORY, *Some polyhedra related to combinatorial problems*, Linear algebra and its applications, 2 (1969), pp. 451–558.

[160] R. E. GOMORY, *Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem*, in 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art, M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, eds., Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 77–103.

[161] P. GOPE AND T. HWANG, *BSN-Care: A secure IoT-based modern healthcare system using body sensor network*, IEEE Sensors Journal, 16 (2016), pp. 1368–1376.

[162] V. GOYAL AND R. RAVI, *A PTAS for the chance-constrained knapsack problem with random item sizes*, Operations Research Letters, 38 (2010), pp. 161–164.

[163] Z. GU, G. L. NEMHAUSER, AND M. W. SAVELSBERGH, *Lifted flow cover inequalities for mixed 0-1 integer programs*, Mathematical Programming, 85 (1999), pp. 439–467.

[164] P. GUPTA AND P. R. KUMAR, *The capacity of wireless networks*, IEEE Transactions on Information Theory, 46 (2000), pp. 388–404.

[165] R. GUPTA, J. MUSACCHIO, AND J. WALRAND, *Sufficient rate constraints for QoS flows in ad-hoc networks*, Ad Hoc Networks, 5 (2007), pp. 429–443.

[166] Y. GUREVICH, L. STOCKMEYER, AND U. VISHKIN, *Solving NP-hard problems on graphs that are almost trees and an application to facility location problems*, Journal of the ACM (JACM), 31 (1984), pp. 459–473.

[167] V. GUSEV, *The vertex cover game: Application to transport networks*, Omega, 97 (2020). 102102.

[168] S. HADDAD-VANIER, C. GICQUEL, L. BOUKHATEM, K. LAZRI, AND P. CHAIGNON, *Virtual network functions placement for defense against distributed denial of service attacks*, in ICORES 2019 - Proceedings of the 8th International Conference on Operations Research and Enterprise Systems, 2019, pp. 142–150.

[169] A. A. HAGBERG, D. A. SCHULT, AND P. J. SWART, *Exploring network structure, dynamics, and function using NetworkX*, tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[170] A. A. HAGBERG, D. A. SCHULT, AND P. J. SWART, *Exploring network structure, dynamics, and function using networkx*, in Proceedings of the 7th Python in Science Conference, G. Varoquaux, T. Vaught, and J. Millman, eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[171] S. HAN, A. GÓMEZ, AND O. A. PROKOPYEV, *Fractional 0–1 programming and submodularity*, Journal of Global Optimization, (2022), pp. 1–17.

[172] I. HARJUNKOSKI, R. NYSTRÖM, AND A. HORCH, *Integration of scheduling and control—theory or practice?*, Computers & Chemical Engineering, 33 (2009), pp. 1909–1918.

[173] I. HARJUNKOSKI, T. WESTERLUND, R. PÖRN, AND H. SKRIFVARS, *Different transformations for solving non-convex trim-loss problems by MINLP*, European Journal of Operational Research, 105 (1998), pp. 594–603.

[174] T. A. HARTMANN, S. LENDL, AND G. J. WOEGINGER, *Continuous facility location on graphs*, Mathematical Programming, 192 (2022), pp. 207–227.

[175] T. HE AND M. TAWARMALANI, *A new framework to relax composite functions in nonlinear programs*, Mathematical Programming, 190 (2021), pp. 427–466.

[176] ——, *Tractable relaxations of composite functions*, Mathematics of Operations Research, 47 (2022), pp. 1110–1140.

[177] J.-B. HIRIART-URRUTY AND C. LEMARÉCHAL, *Fundamentals of convex analysis*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[178] R. HORST AND H. TUY, *Global Optimization: Deterministic Approaches*, Springer, Berlin, 1990.

[179] O. HUDEC, *Confined location of facilities on a graph*, Optimization, 28 (1994), pp. 333–338.

[180] IBM ILOG, *Automatic reformulation of special ordered sets*. https://www.ibm.com/docs/en/icos/20.1.0?topic=sos-automatic-reformulation-special-ordered-sets, 11 2021. [Online; accessed 29-March-2022].

[181] R. A. JABR, *Inductor design using signomial programming*, COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, 26 (2007), pp. 461–475.

[182] P. JACQUET, P. MÜHLETHALER, T. CLAUSEN, A. LAOUITI, A. QAYYUM, AND L. VIENNOT, *Optimized link state routing protocol for ad hoc networks*, in Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International, 2001, pp. 62–68.

[183] P. R. JENKINS, B. J. LUNDAY, AND M. J. ROBBINS, *Robust, multi-objective optimization for the military medical evacuation location-allocation problem*, Omega, 97 (2020), p. 102088.

[184] M. JEPSEN, B. PETERSEN, S. SPOORENDONK, AND D. PISINGER, *Subset-row inequalities applied to the vehicle-routing problem with time windows*, Operations Research, 56 (2008), pp. 497–511.

[185] C. JONCOUR, S. MICHEL, R. SADYKOV, D. SVERDLOV, AND F. VANDERBECK, *Column generation based primal heuristics*, Electronic Notes in Discrete Mathematics, 36 (2010), pp. 695–702.

[186] P. KALCZYNSKI AND Z. DREZNER, *The obnoxious facilities planar p-median problem*, OR Spectrum, 43 (2021), pp. 577–593.

[187] ——, *The obnoxious facilities planar p-median problem with variable sizes*, Omega, 111 (2022), p. 102639.

[188] O. KHAMISOV, *On optimization properties of functions, with a concave minorant*, Journal of Global Optimization, 14 (1999), pp. 79–101.

[189] F. KILINÇ-KARZAN, S. KÜÇÜKYAVUZ, AND D. LEE, *Joint chance-constrained programs and the intersection of mixing sets through a submodularity lens*, Mathematical Programming, (2021), pp. 1–44.

[190] J. M. KLEINBERG, *Single-source unsplittable flow*, in Annual Symposium on Foundations of Computer Science - Proceedings, IEEE, 1996, pp. 68–77.

[191] ——, *Decision algorithms for unsplittable flow and the half-disjoint paths problem*, in Conference Proceedings of the Annual ACM Symposium on Theory of Computing, 1998, pp. 530–539.

[192] G. R. KOCIS AND I. E. GROSSMANN, *Computational experience with dicopt solving minlp problems in process systems engineering*, Computers & Chemical Engineering, 13 (1989), pp. 307–315.

[193] T. G. KOLDA AND B. W. BADER, *Tensor decompositions and applications*, SIAM review, 51 (2009), pp. 455–500.

[194] F. KILINÇ-KARZAN AND S. YILDIZ, *Two-term disjunctions on the second-order cone*, Mathematical Programming, 154 (2015), pp. 463–491.

[195] F. KILINÇ-KARZAN, *On minimal valid inequalities for mixed integer conic programs*, Mathematics of Operations Research, 41 (2016), pp. 477–510.

[196] J. B. LASSERRE, *Semidefinite programming vs. lp relaxations for polynomial programming*, Mathematics of operations research, 27 (2002), pp. 347–360.

[197] J. B. LASSERRE, *An introduction to polynomial and semi-algebraic optimization*, vol. 52, Cambridge University Press, 2015.

[198] A. LAUBE, S. MARTIN, D. QUADRI, AND K. ALAGHA, *Optimal flow aggregation for global energy savings in multi-hop wireless networks*, in Ad-hoc, Mobile, and Wireless Networks, N. Mitton, V. Loscri, and A. Mouradian, eds., Cham, 2016, Springer International Publishing, pp. 124–137.

[199] M. LEITNER, I. LJUBIĆ, M. RIEDLER, AND M. RUTHMAIR, *Exact approaches for network design problems with relays*, INFORMS Journal on Computing, 31 (2019), pp. 171–192.

[200] S. Y. R. LI, R. W. YEUNG, AND N. CAI, *Linear network coding*, IEEE Transactions on Information Theory, 49 (2003), pp. 371–381.

[201] L. LIBERTI, *Writing global optimization software*, in Liberti and Maculan [204], pp. 211–262.

[202] ——, *Spherical cuts for integer programming problems*, International Transactions in Operational Research, 15 (2008), pp. 283–294.

[203] ——, *Undecidability and hardness in mixed-integer nonlinear programming*, RAIRO-Operations Research, 53 (2019), pp. 81–109.

[204] L. LIBERTI AND N. MACULAN, eds., *Global Optimization: from Theory to Implementation*, Springer, Berlin, 2006.

[205] L. LIBERTI AND C. C. PANTELIDES, *Convex envelopes of monomials of odd degree*, Journal of Global Optimization, 25 (2003), pp. 157–168.

[206] F. LIERS, E. MARINARI, U. PAGACZ, F. RICCI-TERSENGHI, AND V. SCHMITZ, *A non-disordered glassy model with a tunable interaction range*, Journal of Statistical Mechanics: Theory and Experiment, 2010 (2010), p. L05003.

[207] M.-H. LIN AND J.-F. TSAI, *Range reduction techniques for improving computational efficiency in global optimization of signomial geometric programming problems*, European Journal of Operational Research, 216 (2012), pp. 17–25.

[208] I. LJUBIĆ, R. WEISKIRCHER, U. PFERSCHY, G. W. KLAU, P. MUTZEL, AND M. FISCHETTI, *An Algorithmic Framework for the Exact Solution of the Prize-Collecting Steiner Tree Problem*, Mathematical Programming, 105 (2006), pp. 427–449.

[209] M. LOCATELLI, *Polyhedral subdivisions and functional forms for the convex envelopes of bilinear, fractional and other bivariate functions over general polytopes*, Journal of Global Optimization, 66 (2016), pp. 629–668.

[210] ——, *Convex envelopes of bivariate functions through the solution of KKT systems*, Journal of Global Optimization, 72 (2018), pp. 277–303.

[211] M. LOCATELLI AND F. SCHOEN, *On convex envelopes for bivariate functions over polytopes*, Mathematical Programming, 144 (2014), pp. 65–91.

[212] L. LOVÁSZ, *Submodular functions and convexity*, in Mathematical programming the state of the art, Springer, 1983, pp. 235–257.

[213] W. LU, Y. GONG, X. LIU, J. WU, AND H. PENG, *Collaborative energy and information transfer in green wireless sensor networks for smart cities*, IEEE Transactions on Industrial Informatics, 14 (2018), pp. 1585–1593.

[214] M. LÜBBECKE AND C. PUCHERT, *Primal heuristics for branch-and-price algorithms*, in Operations Research Proceedings 2011, Springer, 2012, pp. 65–70.

[215] M. E. LÜBBECKE AND J. DESROSIERS, *Selected topics in column generation*, Operations Research, 53 (2005), pp. 1007–1023.

[216] J. LUEDTKE AND S. AHMED, *A sample approximation approach for optimization with probabilistic constraints*, SIAM Journal on Optimization, 19 (2008), pp. 674–699.

[217] J. LUEDTKE, M. NAMAZIFAR, AND J. LINDEROTH, *Some results on the strength of relaxations of multilinear functions*, Mathematical programming, 136 (2012), pp. 325–351.

[218] A. LUNDELL, A. SKJÄL, AND T. WESTERLUND, *A reformulation framework for global optimization*, Journal of Global Optimization, 57 (2013), pp. 115–141.

[219] A. LUNDELL AND T. WESTERLUND, *Solving global optimization problems using reformulations and signomial transformations*, Computers & Chemical Engineering, 116 (2018), pp. 122–134.

[220] C. D. MARANAS AND C. A. FLOUDAS, *Finding all solutions of nonlinearly constrained systems of equations*, Journal of Global Optimization, 7 (1995), pp. 143–182.

[221] ——, *Global optimization in generalized geometric programming*, Computers & Chemical Engineering, 21 (1997), pp. 351–369.

[222] M. MARTENS, F. SALAZAR, AND M. SKUTELLA, *Convex combinations of single source unsplittable flows*, in Algorithms – ESA 2007, L. Arge, M. Hoffmann, and E. Welzl, eds., Berlin, Heidelberg, 2007, Springer Berlin Heidelberg, pp. 395–406.

[223] G. P. MCCORMICK, *Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems*, Mathematical programming, 10 (1976), pp. 147–175.

[224] C. A. MEYER AND C. A. FLOUDAS, *Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes*, Journal of Global Optimization, 29 (2004), pp. 125–155.

[225] C. A. MEYER AND C. A. FLOUDAS, *Trilinear monomials with positive or negative domains: Facets of the convex and concave envelopes*, in Frontiers in Global Optimization, C. A. Floudas and P. Pardalos, eds., Boston, MA, 2004, Springer US, pp. 327–352.

[226] R. MISENER AND C. A. FLOUDAS, *Global optimization of mixed-integer models with quadratic and signomial functions: a review*, Applied and Computational Mathematics, (2012).

[227] ——, *ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations*, Journal of Global Optimization, 59 (2014), pp. 503–526.

[228] ——, *A framework for globally optimizing mixed-integer signomial programs*, Journal of Optimization Theory and Applications, 161 (2014), pp. 905–932.

[229] S. MODARESI, M. R. KILINÇ, AND J. P. VIELMA, *Split cuts and extended formulations for Mixed Integer Conic Quadratic Programming*, Operations Research Letters, 43 (2015), pp. 10–15.

[230] S. MODARESI, M. R. KILINÇ, AND J. P. VIELMA, *Intersection cuts for nonlinear integer programming: convexification techniques for structured sets*, Mathematical Programming, 155 (2016), pp. 575–611.

[231] M. MOZAFFARI, W. SAAD, M. BENNIS, Y. H. NAM, AND M. DEBBAH, *A tutorial on UAVs for wireless networks: Applications, challenges, and open Problems*, IEEE Communications Surveys and Tutorials, 21 (2019), pp. 2334–2360.

[232] G. MUÑOZ, J. PAAT, AND F. SERRANO, *Towards a characterization of maximal quadratic-free sets*, arXiv preprint arXiv:2211.05185, (2022).

[233] G. MUÑOZ AND F. SERRANO, *Maximal quadratic-free sets*, Mathematical Programming, 192 (2022), pp. 229–270.

[234] K. MUROTA, *Discrete convex analysis*, Mathematical Programming, 83 (1998), pp. 313–371.

[235] R. MURRAY, V. CHANDRASEKARAN, AND A. WIERMAN, *Signomial and polynomial optimization via relative entropy and partial dualization*, Mathematical Programming Computation, 13 (2021), pp. 257–295.

[236] G. NEMHAUSER AND L. WOLSEY, *Matroid and Submodular Function Optimization*, John Wiley & Sons, Ltd, 1988, ch. III.3, pp. 659–719.

[237] G. L. NEMHAUSER, L. A. WOLSEY, AND M. L. FISHER, *An analysis of approximations for maximizing submodular set functions—i*, Mathematical programming, 14 (1978), pp. 265–294.

[238] D. NGUYEN, T. TRAN, T. NGUYEN, AND B. BOSE, *Wireless broadcast using network coding*, IEEE Transactions on Vehicular Technology, 58 (2009), pp. 914–925.

[239] T. T. NGUYEN, J.-P. P. RICHARD, AND M. TAWARMALANI, *Deriving convex hulls through lifting and projection*, Mathematical Programming, 169 (2018), pp. 377–415.

[240] B. NI, N. SANTHAPURI, Z. ZHONG, AND S. NELAKUDITI, *Routing with opportunistically coded exchanges in wireless mesh networks*, in 2006 2nd IEEE Workshop on Wireless Mesh Networks, WiMESH 2006, IEEE, 2006, pp. 157–159.

[241] W. NI, J. SHU, M. SONG, D. XU, AND K. ZHANG, *A branch-and-price algorithm for facility location with general facility cost functions*, INFORMS Journal on Computing, 33 (2021), pp. 86–104.

[242] P. OLIVIER, A. LODI, AND G. PESANT, *The quadratic multiknapsack problem with conflicts and balance constraints*, INFORMS Journal on Computing, 33 (2021), pp. 949–962.

[243] M. M. J. OPGENOORD, B. S. COHEN, AND W. W. HOBURG, *Comparison of algorithms for including equality constraints in signomial programming*, ACDL Technical Report TR-2017-1, (2017), pp. 1–23.

[244] O. S. OUBBATI, M. ATIQUZZAMAN, P. LORENZ, M. H. TAREQUE, AND M. S. HOSSAIN, *Routing in flying ad hoc networks: Survey, constraints, and future challenge perspectives*, IEEE Access, 7 (2019), pp. 81057–81105.

[245] U. PAGACZ, *POLIP: Library for polynomially constrained mixed-integer programming*, Jun 2023. Online accessed.

[246] N. R. PAUL, B. J. LUNDAY, AND S. G. NURRE, *A multiobjective, maximal conditional covering location problem applied to the relocation of hierarchical emergency response facilities*, Omega, 66 (2017), pp. 147–158.

[247] D. PECIN, A. PESSOA, M. POGGI, AND E. UCHOA, *Improved branch-cut-and-price for capacitated vehicle routing*, Mathematical Programming Computation, 9 (2017), pp. 61–100.

[248] M. PELEGRÍN AND C. D'AMBROSIO, *Aircraft deconfliction via mathematical programming: Review and insights*, tech. rep., École Polytechnique, Mar. 2021.

[249] M. PELEGRÍN AND L. XU, *Continuous covering on networks: Improved mixed integer programming formulations*, Omega, (2023), p. 102835.

[250] A. PESSOA, R. SADYKOV, AND E. UCHOA, *Solving bin packing problems using vrpsolver models*, in Operations Research Forum, vol. 2, Springer, 2021, p. 20.

[251] A. PESSOA, R. SADYKOV, E. UCHOA, AND F. VANDERBECK, *Automation and combination of linear-programming based stabilization techniques in column generation*, INFORMS Journal on Computing, 30 (2018), pp. 339–360.

[252] A. PESSOA, R. SADYKOV, E. UCHOA, AND F. VANDERBECK, *A generic exact solver for vehicle routing and related problems*, Mathematical Programming, 183 (2020), pp. 483–523.

[253] E. N. PISTIKOPOULOS, A. BARBOSA-POVOA, J. H. LEE, R. MISENER, A. MITSOS, G. V. REKLAITIS, V. VENKATASUBRAMANIAN, F. YOU, AND R. GANI, *Process systems engineering–the generation next?*, Computers & Chemical Engineering, 147 (2021), p. 107252.

[254] F. PLASTRIA, *Continuous covering location problems*, in Facility location: applications and theory, Z.Drezner and H.Hamacher, eds., Springer, 2002, pp. 37–79.

[255] W. H. PRESS, S. A. TEUKOLSKY, W. T. VETTERLING, AND B. FLANNERY, *Chapter 9: Root finding and nonlinear sets of equations*, Book: Numerical Recipes: The Art of Scientific Computing, New York, Cambridge University Press„ 10 (2007).

[256] J. PUCHINGER, G. R. RAIDL, AND U. PFERSCHY, *The multidimensional knapsack problem: Structure and algorithms*, INFORMS Journal on Computing, 22 (2010), pp. 250–265.

[257] J. PUERTO, F. RICCA, AND A. SCOZZARI, *Extensive facility location problems on networks: an updated review*, Top, 26 (2018), pp. 187–226.

[258] H. RAHIMI, A. ZIBAEENEJAD, AND A. A. SAFAVI, *A novel IoT architecture based on 5G-IoT and next generation technologies*, in 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018, IEEE, 2019, pp. 81–88.

[259] F. RENDL, G. RINALDI, AND A. WIEGELE, *Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations*, Mathematical Programming, 121 (2010), pp. 307–335.

[260] J. M. RHYS, *A selection problem of shared fixed costs and network flows*, Management Science, 17 (1970), pp. 200–207.

[261] J.-P. P. RICHARD AND S. S. DEY, *The group-theoretic approach in mixed integer programming*, in 50 Years of Integer Programming 1958-2008, Springer, 2010, pp. 727–801.

[262] A. D. RIKUN, *A convex envelope formula for multilinear functions*, Journal of Global Optimization, 10 (1997), pp. 425–437.

[263] G. RINALDI, *Rudy*, http://www-user. tu-chemnitz. de/helmberg/rudy. tar. gz, (1998).

[264] D. M. RYAN AND B. A. FOSTER, *An integer programming approach to scheduling*, Computer scheduling of public transport urban passenger vehicle and crew scheduling, (1981), pp. 269–280.

[265] A. N. SADIGH, M. MOZAFARI, AND A. H. KASHAN, *A mixed integer linear program and tabu search approach for the complementary edge covering problem*, Advances in Engineering Software, 41 (2010), pp. 762–768.

[266] G. SAGNOL, *Approximation of a maximum-submodular-coverage problem involving spectral functions, with application to experimental designs*, Discrete Applied Mathematics, 161 (2013), pp. 258–276.

[267] G. SAGNOL AND R. HARMAN, *Computing exact d-optimal designs by mixed integer second-order cone programming*, The Annals of Statistics, 43 (2015), pp. 2198–2224.

[268] A. SAXENA, P. BONAMI, AND J. LEE, *Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations*, Mathematical programming, 130 (2011), pp. 359–413.

[269] A. SCHÖBEL, *Locating Dimensional Facilities in a Continuous Space*, Springer International Publishing, Cham, 2019, pp. 143–184.

[270] A. SCHRIJVER ET AL., *Combinatorial optimization: polyhedra and efficiency*, vol. 24, Springer, 2003.

[271] F. SERRANO, *Intersection cuts for factorable MINLP*, in Integer Programming and Combinatorial Optimization, A. Lodi and V. Nagarajan, eds., Cham, 2019, Springer International Publishing, pp. 385–398.

[272] F. SERRANO MUSALEM, *On cutting planes for mixed-integer nonlinear programming*, doctoral thesis, Technische Universität Berlin, Berlin, 2021.

[273] F. S. SHAIKH AND R. WISMÜLLER, *Routing in multi-hop cellular device-to-device (D2D) networks: A survey*, IEEE Communications Surveys & Tutorials, 20 (2018), pp. 2622–2657.

[274] M. SHAMAIAH, S. BANERJEE, AND H. VIKALO, *Greedy sensor selection: Leveraging submodularity*, in 49th IEEE conference on decision and control (CDC), IEEE, 2010, pp. 2572–2577.

[275] P. SHEN, *Linearization method of global optimization for generalized geometric programming*, Applied Mathematics and Computation, 162 (2005), pp. 353–370.

[276] H. D. SHERALI, *Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets*, Acta mathematica vietnamica, 22 (1997), pp. 245–270.

[277] X. SHI, O. A. PROKOPYEV, AND B. ZENG, *Sequence independent lifting for a set of submodular maximization problems*, Mathematical Programming, (2022), pp. 1–46.

[278] S. SHIRINIVAS, S. VETRIVEL, AND N. ELANGO, *Applications of graph theory in computer science an overview*, International Journal of Engineering Science and Technology, 2 (2010), pp. 4610–4621.

[279] O. V. SHYLO, O. A. PROKOPYEV, AND A. J. SCHAEFER, *Stochastic operating room scheduling for high-volume specialties under block booking*, INFORMS Journal on Computing, 25 (2013), pp. 682–692.

[280] Y. SONG, J. R. LUEDTKE, AND S. KÜÇÜKYAVUZ, *Chance-constrained binary packing problems*, INFORMS Journal on Computing, 26 (2014), pp. 735–747.

[281] E. SPEAKMAN AND J. LEE, *Quantifying double mccormick*, Mathematics of Operations Research, 42 (2017), pp. 1230–1253.

[282] M. TAWARMALANI, J.-P. P. RICHARD, AND C. XIONG, *Explicit convex and concave envelopes through polyhedral subdivisions*, Mathematical Programming, 138 (2013), pp. 531–577.

[283] M. TAWARMALANI AND N. V. SAHINIDIS, *A polyhedral branch-and-cut approach to global optimization*, Mathematical Programming, 103 (2005), pp. 225–249.

[284] ——, *A polyhedral branch-and-cut approach to global optimization*, Mathematical programming, 103 (2005), pp. 225–249.

[285] D. E. TILIUTE, *Battery management in wireless sensor networks*, Elektronika ir elektrotechnika, 1 (2007), pp. 9–12.

[286] C. TJANDRAATMADJA, R. ANDERSON, J. HUCHETTE, W. MA, K. K. PATEL, AND J. P. VIELMA, *The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification*, Advances in Neural Information Processing Systems, 33 (2020), pp. 21675–21686.

[287] D. M. TOPKIS, *Supermodularity and complementarity*, Princeton university press, Princeton, 2011.

[288] C. TOREGAS AND C. REVELLE, *Optimal location under time or distance constraints*, Papers of the Regional Science Association, 28 (1972), pp. 131–143.

[289] E. TOWLE AND J. LUEDTKE, *Intersection disjunctions for reverse convex sets*, Mathematics of Operations Research, 47 (2022), pp. 297–319.

[290] C. TSAY, J. KRONQVIST, A. THEBELT, AND R. MISENER, *Partition-based formulations for mixed-integer optimization of trained relu neural networks*, Advances in Neural Information Processing Systems, 34 (2021), pp. 3068–3080.

[291] H. TUY, *Concave programming under linear constraints*, Soviet Mathematics, 5 (1964), pp. 1437–1440.

[292] H. TUY, *Concave minimization under linear constraints with special structure*, Optimization, 16 (1985), pp. 335–352.

[293] P. M. VAIDYA, *A new algorithm for minimizing convex functions over convex sets*, Mathematical programming, 73 (1996), pp. 291–341.

[294] P. H. VANCE, C. BARNHART, E. L. JOHNSON, AND G. L. NEMHAUSER, *Solving binary cutting stock problems by column generation and branch-and-bound*, Computational Optimization and Applications, 3 (1994), pp. 111–130.

[295] F. VANDERBECK, *Implementing mixed integer column generation*, in Column Generation, G. Desaulniers, J. Desrosiers, and M. M. Solomon, eds., Springer US, Boston, MA, 2005, pp. 331–358.

[296] F. VANDERBECK AND L. A. WOLSEY, *Reformulation and Decomposition of Integer Programs*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 431–502.

[297] S. VANIER, *Column generation for the energy-efficient in multi-hop wireless networks problem*, in 16th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, CTW 2018 - Proceedings of the Workshop, 2019, pp. 173–174.

[298] S. VANIER AND K. ARNAUD, *New partition inequalities for the unsplittable flow problem*, in ISCO International Symposium on Combinatorial Optimization, 2018.

[299] J. P. VIELMA, S. AHMED, AND G. NEMHAUSER, *Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions*, Operations Research, 58 (2010), pp. 303–315.

[300] S. VIGERSKE, *MINLPLib: A Library of Mixed-Integer and Continuous Nonlinear Programming Instances*, Feb 2022. [Online; accessed 1. Feb. 2022].

[301] S. WANG, J. LI, AND S. MEHROTRA, *Chance-constrained multiple bin packing problem with an application to operating room planning*, INFORMS Journal on Computing, 33 (2021), pp. 1661–1677.

[302] W. WANG, S. WU, S. WANG, L. ZHEN, AND X. QU, *Emergency facility location problems in logistics: Status and perspectives*, Transportation research part E: logistics and transportation review, 154 (2021), p. 102465.

[303] L. WEI, M. LAI, A. LIM, AND Q. HU, *A branch-and-price algorithm for the two-dimensional vector packing problem*, European Journal of Operational Research, 281 (2020), pp. 25–35.

[304] L. WEI, Z. LUO, R. BALDACCI, AND A. LIM, *A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems*, INFORMS Journal on Computing, 32 (2020), pp. 428–443.

[305] A. WIEGELE, *Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size*, Preprint, 51 (2007).

[306] L. WOLSEY, *Integer Programming*, John Wiley & Sons, 1998.

[307] F. WU, T. WU, AND M. R. YUCE, *An internet-of-things (IoT) network system for connected safety and health monitoring applications*, Sensors, 19 (2019), p. 21.

[308] T. WU, F. WU, J. M. REDOUTE, AND M. R. YUCE, *An autonomous wireless body area network implementation towards IoT connected healthcare applications*, IEEE Access, 5 (2017), pp. 11413–11422.

[309] Y. XIN, X. WANG, G. LEUS, G. YUE, AND J. JIANG, *Interference management in wireless communication systems: Theory and applications*, EURASIP Journal on Wireless Communications and Networking, 2010 (2011).

[310] G. XU, *Global optimization of signomial geometric programming problems*, European journal of operational research, 233 (2014), pp. 500–510.

[311] L. XU, *Optimal location of safety landing sites*, tech. rep., LIX, École Polytechnique, 2021. hal-03286640.

[312] L. XU, C. D'AMBROSIO, L. LIBERTI, AND S. H. VANIER, *On cutting planes for extended formulation of signomial programming*, 2022.

[313] L. XU AND S. HADDAD VANIER, *Branch-and-price for energy optimization in multi-hop wireless sensor networks*, Networks, 80 (2022), pp. 123–148.

[314] Y. YANG AND R. KRAVETS, *Contention-aware admission control for ad hoc networks*, IEEE Transactions on Mobile Computing, 4 (2005), pp. 363–377.

[315] I. YAQOOB, I. A. T. HASHEM, Y. MEHMOOD, A. GANI, S. MOKHTAR, AND S. GUIZANI, *Enabling communication technologies for smart cities*, IEEE Communications Magazine, 55 (2017), pp. 112–120.

[316] Q. YU AND S. KÜÇÜKYAVUZ, *Strong valid inequalities for a class of concave submodular minimization problems under cardinality constraints*, Mathematical Programming, (2023), pp. 1–59.

[317] S. ZHANG, J. S. C. SALAZAR, C. FELDMANN, D. WALZ, F. SANDFORT, M. MATHEA, C. TSAY, AND R. MISENER, *Optimizing over trained gnns via symmetry breaking*, arXiv preprint arXiv:2305.09420, (2023).

[318] Y. ZHANG, R. JIANG, AND S. SHEN, *Ambiguous chance-constrained binary programs under mean-covariance information*, SIAM Journal on Optimization, 28 (2018), pp. 2922–2944.

[319] Z. ZHANG, B. T. DENTON, AND X. XIE, *Branch and price for chance-constrained bin packing*, INFORMS Journal on Computing, 32 (2020), pp. 547–564.

ECOLE DOCTORALE

Titre: Méthodes de relaxation pour la programmation non linéaire en nombres entiers mixtes

Mots clés: PNLNE; optimisation globale ; relaxation

Résumé: Cette thèse se concentre sur la programmation non linéaire à variables mixtes (MINLP), une classe de problèmes d'optimisation mathématique, et les algorithmes associés pour les résoudre. L'algorithme central utilisé dans de nombreux solveurs d'optimisation globale pour les problèmes MINLP est l'algorithme de séparation et évaluation. La clé du succès de l'algorithme de séparation et évaluation réside dans l'utilisation de relaxations des problèmes d'optimisation, qui sont essentielles pour obtenir des bornes duales efficaces. Cependant, la construction de relaxations efficaces dépend des structures spécifiques des problèmes d'optimisation. Dans la première partie de cette thèse, nous présentons un aperçu complet des outils de relaxation structurelle adaptés aux problèmes MINLP structurés liés à différents domaines d'applications. Ces outils englobent des relaxations à partir de formulations étendues, des relaxations par sous-modularité, des relaxations utilisant une approximation linéaire par morceaux et des renforcements de relaxation via des coupes d'intersection. Nous développons de nouveaux résultats théoriques avancés basés sur ces outils. Dans la deuxième partie, nous utilisons ces techniques de relaxation pour aborder divers problèmes d'optimisation. Nous explorons les plans coupants pour la programmation signoïdale. Nous proposons des coupes d'intersection pour améliorer les relaxations linéaire des problèmes d'optimisation sous-modulaire. Nous étudions les relaxations de Dantzig-Wolfe pour un problème de programmation linéaire à variables mixtes dans le routage de réseaux sans fil et un problème MINLP dans le bin-packing sous-modulaire. Enfin, nous étudions la technique de relaxation big-M appliquée aux fonctions linéaires par morceaux dans le problème de couverture continue sur un réseau. Les travaux réalisés durant cette thèse de doctorat contribuent à l'avancement des approches de la programmation non linéaire en nombre entiers et des méthodes d'optimisation connexes. En effet la combinaison des études exhaustives réalisées sur diverses techniques de relaxation et leurs applications à différents contextes d'optimisation offrent des perspectives précieuses tant pour la compréhension théorique des problèmes que pour la mise en œuvre empirique des résultats.

Title: Relaxation methods for mixed-integer nonlinear programming

Keywords: MINLP; global optimization; relaxation

Abstract: This thesis focuses on mixed-integer nonlinear programming (MINLP), a class of mathematical optimization problems, and the associated algorithms to solve them. The core algorithm utilized in many global optimization solvers for MINLP problems is the branch-and-bound algorithm. Key to the success of the branch-and-bound approach is the use of relaxations of optimization problems, which are vital in obtaining efficient and tight dual bounds. However, constructing effective relaxations depends on the specific structures of optimization problems. In the first part of this thesis, we present a comprehensive overview of structural relaxation tools tailored for structured MINLP problems across different disciplines. These tools encompass relaxations from extended formulations, relaxations via submodularity, relaxations using piece-wise linear approximation, and relaxation tightening via intersection cuts. Then, we develop novel advanced theoretical results based on these tools. In the second part, we employ these relaxation techniques to address various optimization problems. We explore cutting planes for signomial programming. Then, we propose intersection cuts for enhancing linear programming relaxations of submodular optimization problems. Next, we investigate the Dantzig-Wolfe relaxations for a mixed-integer linear programming problem in wireless network routing and a MINLP problem in submodular bin-packing. Finally, we study the big-M relaxation technique as applied to piece-wise linear functions in the continuous covering problem on a network. By combining these comprehensive studies on various relaxation techniques and their applications in different optimization contexts, this thesis contributes to the advancement of MINLP and related optimization methods, offering valuable insights for both theoretical understanding and computational implementation.

Institut Polytechnique de Paris
91120 Palaiseau, France