



Titre: Data-driven decomposition and constraint customization for
Title: combinatorial optimization

Auteur: Mahdis Bayani
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Bayani, M. (2023). Data-driven decomposition and constraint customization for
Citation: combinatorial optimization [Thèse de doctorat, Polytechnique Montréal].
PolyPublie. <https://publications.polymtl.ca/55112/>

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

URL de PolyPublie: <https://publications.polymtl.ca/55112/>
PolyPublie URL:

**Directeurs de
recherche:** Louis-Martin Rousseau, & Yossiri Adulyasak
Advisors:

Programme: Doctorat en mathématiques
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Data-driven decomposition and constraint customization for combinatorial
optimization**

MAHDIS BAYANI

Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Août 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Cette thèse intitulée :

**Data-driven decomposition and constraint customization for combinatorial
optimization**

présentée par **Mahdis BAYANI**

en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Michel GENDREAU, président

Louis-Martin ROUSSEAU, membre et directeur de recherche

Yossiri ADULYASAK, membre et codirecteur de recherche

Andrea LODI, membre

Mike HEWITT, membre externe

DEDICATION

*To Peyman,
whose endless love and belief in me made my PhD journey more exciting.*

*To my parents,
who always supported my decisions and without their sacrifices this could not happen.*

*To my brother and sister,
who gave me motivation and support every single day.*

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Prof. Louis-Martin Rousseau, and Prof. Yossiri Adulyasak for their unwavering support and motivation throughout my Ph.D. journey. Their insightful feedback and guidance pushed my thinking and brought my work to a higher level. I thank them first for believing in me and accepting me as their student, and also for everything I learned throughout these years in Montreal.

I am also grateful to Prof. Borzou Rostami at University of Alberta who was my co-supervisor for several years. The experiences I obtained while working with him are unique. He always supported me with his invaluable advice.

In addition, I would like to thank professors Michel Gendreau, Andrea Lodi, and Mike Hewitt for accepting to be members of my dissertation committee.

I would like to offer my special thanks to the operations research team and data science team of JDA software company who made the two first years of my PhD precious.

And last but not least, I would like to thank my friends at HANALOG, CIRRELT and GERAD. Many thanks to Adrien, Amir, Hamed, Mahsa, Maryam, Narges, Pedram and Sajad for making my Ph.D. journey more exciting. I want to thank other friends all around the world, Elham, Mohammad, Parinaz, Sahar and Shokufeh for their friendship and support during the years.

RÉSUMÉ

Récemment, on a assisté à une montée en puissance des méthodes basées sur les données dans le domaine de l’optimisation. En général, la construction de modèles mathématiques pour un système implique d’obtenir des informations sur le système pour le modéliser. D’une part, l’émergence du big data motive l’utilisation des méthodes basées sur les données pour trouver les modèles latents des données, estimer les paramètres et construire des modèles d’optimisation. D’autre part, comme la disponibilité des données conduit à des modèles d’optimisation plus sophistiqués, il est nécessaire de développer des méthodes basées sur les données pour s’attaquer à la complexité de résolution de ces problèmes d’optimisation. Certaines études dans ce domaine s’appuient sur des techniques axées sur les données pour évaluer les paramètres des problèmes d’optimisation afin de détecter automatiquement les modèles sous-jacents. Elles tentent d’explorer la structure de la matrice des coûts ou de la matrice des contraintes et de trouver des modèles dans une classe spécifique de problèmes. L’utilisation de l’apprentissage automatique pour traiter directement les instances d’un problème d’optimisation est une autre application des méthodes basées sur les données dans le domaine de l’optimisation. La résolution de modèles d’optimisation par des approches d’apprentissage automatique et l’incorporation du problème d’optimisation dans l’étape d’apprentissage sont quelques exemples dans ce domaine. Dans cette thèse, nous tirons parti des techniques basées sur les données pour étudier et développer des méthodes de résolution de certains problèmes d’optimisation difficiles à résoudre.

Les méthodes de décomposition sont utilisées dans la littérature de l’optimisation depuis de nombreuses années comme une alternative pour traiter les problèmes de grandes dimensions. La décomposition consiste à diviser un problème en sous-problèmes plus petits qui peuvent être résolus séparément. Dans deux chapitres de ce travail, nous étudions la décomposabilité de certains problèmes de Programmation Quadratique Binaire avec des contraintes linéaires par le biais d’une approche basée sur des méthodes basées sur les données. Nous étudions la structure des données relatives aux coûts quadratiques des Programmes Quadratiques Binaires et exploitons les modèles spécifiques liés au type de problème pour reformuler et convexifier les problèmes, et utiliser la décomposition pour les résoudre. La Programmation Quadratique Binaire est une classe de problèmes Optimisation Combinatoire avec des variables binaires et des fonctions objectifs quadratiques.

Dans la première étude de la thèse, nous étudions la généralisation de la “reformulation en étoile” introduite pour le Problème d’Arbre Quadratique Adjacent Minimal dans la littéra-

ture pour d'autres problèmes de Programmation Quadratique Binaire simplement adjacents. Nous considérons deux problèmes de cette classe, le Problème d'Assignment Adjacent Semi Quadratique et le problème de Suivi d'Objets Multiples en vision artificielle pour démontrer l'efficacité de la reformulation en étoile.

La deuxième étude de cette thèse est consacré à la proposition d'un cadre unifié combinant la reformulation en étoile et une approche de division des coûts pour reformuler les problèmes de Programmation Quadratique Binaire généraux. Dans cette approche, nous exploitons la structure du coût quadratique basée sur les coûts adjacents et les coûts non adjacents et fournissons une reformulation. Ainsi, nous tirons parti de la structure en étoile de la nouvelle reformulation pour développer un algorithme de Génération de Colonnes basé sur la décomposition. La composante quadratique du problème est traitée dans le problème principal de Génération de Colonnes et dans ses sous-problèmes.

Dans nos expérimentations, nous évaluons les performances de la méthodologie sur différentes applications avec différentes structures quadratiques. La reformulation en étoile et la Génération de Colonnes sont plus performants que GUROBI en termes de borne duale et de temps de calcul dans presque tous les cas de problèmes de Programmation Quadratique Binaire adjacents. En outre, même une simple mise en œuvre du cadre de fractionnement des coûts donne des résultats encourageants pour le Problème d'Assignment Semi Quadratique en tant qu'étude de cas, en particulier pour les instances avec des matrices quadratiques plus éparses.

Dans le troisième essai, une approche basée sur les données est utilisée pour personnaliser les contraintes des modèles de Programmation Linéaire en Nombres Entiers Mixtes classiques afin de produire une solution qui est probablement réalisable en pratique sur la base des contraintes apprises à partir des solutions précédemment mises en œuvre. Les décideurs de différents secteurs utilisent des logiciels d'optimisation pour la planification et la prise de décision. Cependant, dans de nombreuses applications réelles, ils personnalisent les solutions fournies par le logiciel sur la base de certaines règles et préférences opérationnelles internes implicites. Pour s'assurer que de telles considérations et contraintes cachées peuvent être systématiquement prises en compte sans nécessiter de connaissances explicites obtenues auprès d'experts humains, il est possible d'exploiter des méthodes basées sur les données pour intégrer des contraintes latérales dans des Programmes Linéaires en Nombres Entiers Mixtes classiques. Plutôt que de s'appuyer sur un simple modèle de régression linéaire proposé dans la littérature, nous étendons le cadre en incorporant la régression linéaire régularisée et les arbres de décision dans le modèle d'optimisation. Pour motiver la méthode, nous discutons de différentes combinaisons de règles dans le problème du sac à dos linéaire et binaire. En outre, nous présentons une approche pour traiter les cas programmation linéaire en nombres

entiers mixtes 0-1 en intégrant un arbre de classification au lieu d'une régression linéaire. Nous abordons également des règles opérationnelles plus difficiles et utilisons le cadre dans une application pratique, le Problème de Roulement des Infirmières. Les solutions optimales obtenues par la méthode suggèrent que l'approche proposée peut apprendre les ajustements à partir des données et être plus efficace que les travaux antérieurs dans ce domaine dans des environnements complexes.

MOTS-CLÉS : programmation quadratique binaire, optimisation combinatoire, génération de colonnes, problème de semi-affectation, problème de suivi d'objets multiples, personnalisation des contraintes, méthodes basées sur les données, programmation linéaire en nombres entiers mixtes, arbre de décision.

ABSTRACT

Recently there has been a surge to benefit from data-driven methods in the field of optimization. Generally, constructing mathematical models for a system involves obtaining information from the system to model it. On the one hand, the emergence of big data motivates utilizing data-driven methods to find the latent patterns of the data, estimate the parameters, and build optimization models. On the other hand, since data availability leads to more sophisticated optimization models, it needs to develop data-driven methods to tackle the complexity of solving these optimization problems. Some of the studies in this field leverage data-driven techniques to appraise the parameters of the optimization problems to detect the underlying patterns automatically. They attempt to explore the structure of the cost matrix or constraint matrix and to find patterns in a specific class of problems. Using machine learning to address instances of an optimization problem directly is another application of data-driven methods in optimization. Solving optimization models through machine learning approaches and incorporating the optimization problem into the training step are some examples in this field. In this dissertation, we take advantage of data-driven techniques to study and develop solution methods to tackle some hard-to-solve optimization problems.

Decomposition methods have been used in the optimization literature for many years as an alternative to address large-scale problems. Decomposition revolves around breaking a problem into smaller subproblems that are separately solvable. In two chapters of this work, we study the decomposability of some Binary Quadratic Programming (BQP) problems with linear constraints through a data-driven approach. We investigate the structure of the quadratic cost data of the BQPs and leverage the specific patterns related to the problem type to reformulate and convexify the problems, and use decomposition to solve them. BQP is a class of Combinatorial Optimization (CO) problems with binary variables and quadratic objective functions.

In the first essay of the dissertation, we investigate the generalizability of the star-reformulation introduced for the Adjacent-only Quadratic Minimum Spanning Tree Problem (AQMSTP) in the literature for other adjacent-only BQP problems. We consider two problems of this class, the Adjacent-only Quadratic Semi-Assignment Problem (AQSAP) and the Multiple Object Tracking (MOT) problem in computer vision to demonstrate the efficiency of the star-based framework.

The second essay of this dissertation is dedicated to suggesting a unified framework combining

the star-reformulation and a cost-splitting approach to reformulate general BQP problems. In this approach, we exploit the structure of the quadratic cost based on the adjacent and non-adjacent costs and provide a reformulation. Hence, we leverage the star-based structure of the new reformulation to develop a decomposition-based Column Generation (CG) algorithm. The quadratic component of the problem is dealt with in the CG master problem and its subproblem.

Our computational experiments evaluate the methodology’s performance on different applications with different quadratic structures. The star-reformulation and CG outperform GUROBI in terms of both dual bound and computational time in almost all instances of the adjacent-only BQP problems. Moreover, even a simple implementation of the cost-splitting framework demonstrates encouraging outcomes for Quadratic Semi-Assignment Problem (QSAP) as a case study, particularly for instances with sparser quadratic matrices.

In the third essay, a data-driven approach is utilized to customize the constraints of classical Mixed Integer Linear Programming (MILP) models to produce a solution that is likely practically feasible based on the constraints learned from previously implemented solutions. Decision-makers in different industries use optimization software for planning and decision-making. However, in many real applications, they customize the solutions provided by the software based on some implicit internal operational rules and preferences. To ensure that such considerations and hidden constraints can be systematically considered without requiring explicit knowledge from human experts, one can leverage data-driven methods to embed side constraints in known classical MILPs. Rather than relying on a simple linear regression model proposed in the literature, we extend the framework by incorporating regularized linear regression and decision trees in the optimization model. To motivate the method, we discuss different combinations of rule settings in the linear and binary knapsack problem. Moreover, we demonstrate how we handle 0-1 MILP cases by embedding a classification tree instead of linear regression. We also address more challenging operational rules and employ the framework in a more realistic application, the Nurse Rostering Problem. The optimal solutions obtained by the method suggest that the proposed approach could learn the adjustments from the data and perform more effectively compared to the previous work in this area in complex settings.

KEYWORDS: {binary quadratic programming, combinatorial optimization, column generation, semi-assignment problem, multiple object tracking problem, constraint customization, data-driven methods, mixed integer linear programming, decision tree}

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	viii
TABLE OF CONTENTS	x
LIST OF TABLES	xiii
LIST OF FIGURES	xv
LIST OF SYMBOLS AND ACRONYMS	xvi
LIST OF APPENDICES	xvii
CHAPTER 1 INTRODUCTION	1
1.1 Binary Quadratic Programming	2
1.2 Model Customization in Optimization Problems	3
1.3 Organization of the Dissertation	4
CHAPTER 2 LITERATURE REVIEW	6
2.1 Adjacent-only BQP Problems	6
2.2 BQP Exact Solution Methods	7
2.2.1 Linearization Techniques	8
2.2.2 Decomposition Techniques	9
2.2.3 Other Approaches	9
2.3 Data-driven Methods in Optimization	10
2.3.1 Structure-level	10
2.3.2 Instance-level	11
CHAPTER 3 STAR-REFORMULATION FOR ADJACENT-ONLY BINARY QUADRATIC PROGRAMMING PROBLEMS	16
3.1 Introduction	16

3.1.1	Main Contributions	17
3.2	Star-Reformulation For Adjacent-only BQP Problems	17
3.2.1	Definitions	18
3.2.2	Star-Reformulation	19
3.3	Column Generation Approach	20
3.4	Illustrative Examples	22
3.4.1	Adjacent-only Quadratic Semi-Assignment Problem	22
3.4.2	Multiple Object Tracking	24
3.5	Computational Experiments	29
3.5.1	Adjacent-only QSAP Experiments	31
3.5.2	MOT Experiments	34
3.6	Conclusion	37
CHAPTER 4 A DUAL BOUNDING FRAMEWORK THROUGH COST-SPLITTING FOR BINARY QUADRATIC COMBINATORIAL OPTIMIZATION		38
4.1	Introduction	38
4.1.1	Main Contributions	38
4.2	Cost-Splitting and Star-Reformulation for General BQP Problems	39
4.2.1	Column Generation Approach	40
4.3	Computational Studies	41
4.3.1	Quadratic Semi-Assignment Problem	42
4.3.2	Experiments	44
4.3.3	Results	46
4.4	Conclusion	51
CHAPTER 5 A DATA-DRIVEN METHOD FOR CONSTRAINT CUSTOMIZATION IN OPTIMIZATION MODELS		52
5.1	Introduction	52
5.1.1	Main Contributions	53
5.2	Description of the Solution Framework	53
5.2.1	Regularized Linear Regression	56
5.2.2	Decision Trees	57
5.3	Test Problems and Computational Experiments	59
5.3.1	Linear and Binary Knapsack problems	59
5.3.2	Nurse Rostering Problem	64
5.4	Conclusion	70

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS	72
6.1 Summary of Works	72
6.2 Limitations and Future Research	73
REFERENCES	75
APPENDICES	87

LIST OF TABLES

Table 5.1	Rule settings for knapsack problem	61
Table 5.2	Functional mapping of rules for knapsack problem	61
Table 5.3	Results of applying the framework on different settings for the linear knapsack problem	63
Table 5.4	Results of applying the framework on different settings for 0-1 knapsack problem	64
Table 5.5	Rule setting for Nurse Rostering Problem	67
Table 5.6	Definition of computed gaps for Nurse Rostering Problem	69
Table 5.7	Results of applying the framework on different settings for Nurse Rostering Problem – $\alpha = 3, \alpha' = 1$	70
Table 5.8	Results of applying the framework on different settings for Nurse Rostering Problem – $\alpha = \alpha' = 1$	70
Table A.1	AQSAP – Comparing different methods – At least one of the methods stops within the time limit	89
Table A.2	AQSAP – Comparing different methods – None of the methods stop within the time limit	90
Table A.3	Data association on MOT16-09 data set – Comparing GUROBI and CG – At least one of the methods stops within the time limit	91
Table A.4	Data association on MOT16-09 data set – Comparing GUROBI and CG – None of the methods stops within the time limit	92
Table B.1	QSAP – 10% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit . .	94
Table B.2	QSAP – 10% out-of-star quadratic matrix density – Comparing different methods – None of the methods stops within the time limit	95
Table B.3	QSAP – 15% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit . .	96
Table B.4	QSAP – 15% out-of-star quadratic matrix density – Comparing different methods – None of the methods stops within the time limit	97
Table B.5	QSAP – 20% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit . .	98
Table B.6	QSAP – 20% out-of-star quadratic matrix density – Comparing different methods – None of the methods stops within the time limit	99

Table B.7	QSAP – 25% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit . .	100
Table B.8	QSAP – 25% out-of-star quadratic matrix density – Comparing different methods – None of the methods stops within the time limit	101
Table C.1	MIP constraint representation of rules for Nurse Rostering Problem .	102

LIST OF FIGURES

Figure 3.1	Two examples of graph to demonstrate the concepts of set A , adjacency set $\mathcal{A}$ and star s	18
Figure 3.2	Graphical representation of a solution of data association in MOT applied to detections of four video frames	26
Figure 3.3	Performance profiles for AQSAP instances.	33
Figure 3.4	Performance profiles for the MOT problem	36
Figure 4.1	QSAP-10% out-of-star density	47
Figure 4.2	QSAP-15% out-of-star density	48
Figure 4.3	QSAP-20% out-of-star density	49
Figure 4.4	QSAP-25% out-of-star density	50
Figure 5.1	Framework Overview	56
Figure 5.2	Effect of coefficient c on the quality of the solutions of eNet linear regression on the linear knapsack problem.	62
Figure A.1	Comparing methods in terms of obtaining BFS	90

LIST OF SYMBOLS AND ACRONYMS

AOP	Adopted Optimization Problem
AQMSTP	Adjacent-only Quadratic Minimum Spanning Tree Problem
AQSAP	Adjacent-only Quadratic Semi-Assignment Problem
BBD	Bordered Block Diagonal
BFS	Best-known Feasible Solution
BQP	Binary Quadratic Programming
CG	Column Generation
CO	Combinatorial Optimization
CP	Constraint Programming
DP	Dynamic Programmin
DW	Dantzig-Wolfe
EML	Empirical Model Learning
GOP	General Optimization Problem
LSTM	Long-Short Term Memory
LB	Lower Bound
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
MINLP	Mixed Integer Non-linear Programming
MOT	Multiple Object Tracking
NLP	Nonlinear Programming
NRP	Nurse Rostering Problem
QAP	Quadratic Assignment Problem
QMSTP	Quadratic Minimum Spanning Tree Problem
QSAP	Quadratic Semi-Assignment Problem
RLT	Reformulation Linearization Technique
RMP	Restricted Master Problem
SDP	Semi-Definite Programming
SLT	Standard Linearization Technique
TSP	Traveling Salesman Problem
UB	Upper Bound
UBQP	Unconstrained Binary Quadratic Programming

LIST OF APPENDICES

Appendix A	SUPPLEMENTS OF CHAPTER 3	87
Appendix B	SUPPLEMENTS OF CHAPTER 4	93
Appendix C	SUPPLEMENTS OF CHAPTER 5	102

CHAPTER 1 INTRODUCTION

Combinatorial optimization (CO) problems arise in various fields, including operations research, computer science, and discrete mathematics. Many problems in the domains of routing, scheduling, telecommunication, transportation, planning, and decision-making are categorized as CO problems. Typically, CO problems involve discrete and often combinatorial structures, such as graphs, sets, permutations, and partitions. Examples of CO problems include the Traveling Salesman Problem (TSP), the knapsack problem, the job scheduling problem, and the graph coloring problem [1]. Solving these problems is challenging because many of them are known to be NP-hard or NP-complete, which means that they cannot be solved efficiently by exact algorithms for large-scale instances [2]. While several algorithms have been proposed to solve CO problems, they face limitations when applied to large-scale problems, such as expensive execution in terms of infrastructure or time and lack of generalization due to being specific to particular problem configurations. Specifically, dealing with the cases where the CO problem includes a type of non-linearity either in the objective function or in the constraint is more difficult.

Recently, advances in big data analysis have captured the attention of different scientific communities. Mathematical programming (optimization) is one of the fields that is largely affected by the emergence of big data and the development of techniques to process them. On one hand, mathematical models and optimization algorithms are used to delve into the data and uncover meaningful patterns. Enhancement of using data analysis in real-life applications leads to more challenging and more complex optimization problems derived from them [3, 4]. On the other hand, the fast-growing availability of data and data-driven techniques have a significant role in the development of the optimization field. Constructing optimization models involves utilizing some information and estimations about the parameters of the model as well as the relations between the variables. Therefore, mathematical programming and optimization fields can benefit from data-driven methods to provide more accurate models, and consequently to generate informed decisions from the available data [5]. Furthermore, there is a recent trend of practically solving large-scale optimization problems that is associated with utilizing data-driven methods. These methods aim to identify patterns and structures within the problem data to achieve efficient solutions [6].

One of the relevant ways to deal with the challenges of CO is to employ decomposition methods by breaking large-scale problems into many smaller and more manageable problems. If a problem shows decomposability, it is more likely to provide solutions in less computational

time. Hence, identifying decomposable structures within objective functions or constraints of optimization models is advantageous [7].

The current dissertation explores a data-driven approach as an alternative means of addressing real-life large-scale optimization problems that traditional optimization techniques struggle to solve. Inspired by Khaniyev [6], our investigation digs into two different levels at which data-driven approaches can be applied to optimize problem-solving.

The first level which is the highest level is exploiting the model structure and problem type. Certain classes of optimization problems are known to be efficiently solvable by exploiting special structures embedded in their constraint or objective function matrices. As previously mentioned, decomposability is one of these structures. A problem that represents this structural characteristic has the potential to be solved by decomposition methods. In a part of this dissertation, we aim to leverage data-driven methods to detect the underlying decomposability of Binary Quadratic Programming problems which are normally computationally challenging combinatorial problems.

The second level which we are investigating is the instance level where the instance-specific data is explored for patterns that would help the solution of other problem instances. To this end, in a part of this dissertation, we focus on using data-driven methods to customize the set of constraints in some classic optimization models. As such, we ensure that business-specific rules and preferences are considered in the optimization models without requiring explicit knowledge obtained from human experts.

In this dissertation, we consider two different streams of research: (i) data-driven reformulations applied to solve combinatorial optimization problems where parts of the reformulations are constructed based on the characteristics of the problem and parameters of the instance and (ii) data-driven constraint customization where constraints generated using machine learning models are embedded in an optimization model to determine solutions which are likely feasible considering hidden operational constraints and preferences. The problems and techniques are introduced in the remainder of this chapter and the next one.

1.1 Binary Quadratic Programming

BQP is a large class of Combinatorial Optimization problems that arise from modeling real-life applications, for example, in management, engineering, logistics, and network design [8]. Given graph $G = (V, E)$ with node set $V = \{1, 2, \dots, |V|\}$ and edge set $E = \{1, 2, \dots, m\}$, a BQP problem with linear constraints on graph G can be specified using a quadratic cost

matrix $\mathbf{q} \in \mathbb{R}^{m \times m}$ and a linear cost vector $\mathbf{c} \in \mathbb{R}^m$ and is formulated as follows:

$$\begin{aligned} \text{BQP: } \min \quad & \sum_{e \in E} c_e x_e + \sum_{(e,f) \in \mathcal{E}} q_{ef} x_e x_f \\ \text{s.t. } \quad & \mathbf{x} \in X, \end{aligned} \tag{1.1}$$

where $X \subseteq \{0,1\}^m$ is the set of feasible binary vectors and $\mathcal{E} = \{(e, f) \in E \times E : e < f\}$. Without loss of generality, we only consider the symmetric quadratic costs in the objective function as one can replace q_{ef} and q_{fe} by their average $(q_{ef} + q_{fe})/2$ for the case of asymmetric quadratic costs. Note that the matrices and vectors are shown in bold text throughout the study.

Many quadratic CO problems can be naturally formulated in this fashion. Some important examples include the Quadratic Assignment Problem (QAP) [9], the quadratic knapsack problem [10], the quadratic TSP [11, 12, 13], the quadratic shortest path problem [14, 15], the Quadratic Minimum Spanning Tree Problem (QMSTP) [16, 17, 18], and the quadratic set covering problem [19, 8].

Adjacent-only BQP Problems

In many real-life applications modeled as BQP on graphs, the quadratic costs appear only between adjacent edges. As such, interaction costs are zero for pairs of edges that do not share a common endpoint. Some notable examples include the AQMSTP [20, 18], the quadratic Traveling Salesman Problem [11, 13], the adjacent QAP [21], the adjacent quadratic shortest path problem [22, 14] and variants of the correlation clustering and the modularity maximization problems [23, 24, 25].

Motivated by the issues and difficulties of solving BQPs, in two chapters of this dissertation, we examine a reformulation and lower bounding framework for adjacent-only class of problems and for general BQPs. The reformulations are provided based on the concept of star in graph theory, while the structure of the problems are studied and a decomposition method is used to solve the problems.

1.2 Model Customization in Optimization Problems

Decision-making is about finding the optimal decisions from candidate decisions, which is an optimization problem in essence. Optimization models are employed in industry mostly as software-based decision support systems to assist planners determine actions. The optimization models included in the software are normally traditional models in the most common

formats to be used in many applications in different organizations as the customers of such software. These models rarely reflect the reality of the specific operational settings of the different individual organizations. Hence, there are often some modifications of optimal solutions to adjust the solutions with the internal operational rules and preferences of the customers. These adjustments are mostly human-driven and susceptible to error. They require more effort and planning on top of the already complex and possibly exhaustive optimization models. In addition, since the business rules are possible to alter over time and based on different experts' preferences within their organizations, the customization process is time-consuming and may require to be repeated using multitudes of diverse settings. The main drawback of this type of customization is that the software transforms into a customized decision support system for each individual customer, which is not an ideal solution approach in today's information technology industry [26].

One beneficial strategy is to have a common code-based decision support software for all customers, while the stored data may differ from one customer to the other. The customization of this software is thereby implemented based on the data of each customer. A data-driven approach can be utilized to extract the hidden business rules and preferences of each customer. This way, although the software provider creates only one core model for all organizations, the data-driven customization is applied without any human interference to extend the primary fixed optimization model based on the requirements of each company. By adopting this approach, one will alleviate the critical issues pointed out previously. As such, the solution strategy will be more practically feasible considering the different views of different decision-makers and eventually will be proactive to provide better, more acceptable solutions based on the needs of the organizations [26]. In the last essay of this study, by building upon previous research, we propose a framework to indirectly learn such rules from the historical data of executed plans by customers and to systematically derive solutions that are likely feasible with respect to such hidden rules.

1.3 Organization of the Dissertation

The study presented in this dissertation has yielded two scientific papers as followed:

- Bayani, M., Rostami, B., Adulyasak, Y., Rousseau, LM. A Dual Bounding Framework for Binary Quadratic Combinatorial Optimization, Les Cahiers du GERAD ISSN. 2021 Aug;711:2440. Submitted to INFORMS Journal on Computing in July 2021 (Under 3rd revision).
- Bayani, M., Adulyasak, Y., Rousseau, LM. A data-driven method for constraint cus-

tomization in optimization models (will be submitted to INFORMS Journal On Optimization)

At the beginning of each chapter, we provide some details on the sections which are included in the related articles. The remainder of this dissertation is organized as follows. In chapter 2, we review some of the studies related to the BQP problems and data-driven methods in optimization. In Chapter 3, we examine star-reformulation for adjacent BQP problems. The next chapter is dedicated to proposing the star-reformulation for general BQP problems. In Chapter 5, a data-driven method is suggested to customize the side constraints of MILP problems. Eventually, we conclude the dissertation in Chapter 6.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we review some recent advances in BQP, as well as data-driven methods for optimization.

In 2.1, we provide a description of adjacent-only BQP problems. In Section 2.2, we briefly introduce different exact methods that have been used in the literature to solve the BQP problems. In the last section, we review some of the recent studies in data-driven methods at both the structural level and instance level. We then highlight the gaps in the literature on these topics which lead to our contributions in this study.

2.1 Adjacent-only BQP Problems

The adjacent BQP problem is a special case of BQPs in which we assume zero interaction costs for non-adjacent pairs of edges. Accordingly, only quadratic terms of the form $x_{ij}x_{kl}$ with $j = k$ and $i \neq l$ or with $j \neq k$ and $i = l$ have objective function coefficients that are not zero. In the rest of this section, we explore some of the studies carried out on adjacent BQP problems in the literature.

Variants of the quadratic shortest path problem are investigated in modeling different applications of network optimization. Hence, one of the most studied adjacent-only BQP problems is the adjacent quadratic shortest path problem. Rostami et al. [15] study the complexity of this problem and provide an Lower Bound (LB) based on Gilmore-Lawler bound and improve this obtained LB through an iterative method. In Hu and Sotirov [14], the authors investigate the polynomially solvable cases of quadratic shortest path and adjacent quadratic shortest path. They prove that some special structures in the quadratic cost matrix of the problem lead to solving the associated problem effectively. Recently, an extension of Dijkstra's algorithm is proposed for adjacent quadratic shortest path problems in Vilela et al. [27]. They show that the algorithm can solve the problem in polynomial time under some mild assumption. Moreover, they suggest a labeling algorithm to speed up the search for solutions to this problem.

Another adjacent problem that is studied in the literature is the adjacent QAP. In Rostami et al. [28], two special Reformulation Linearization Technique (RLT) formulations are proposed to obtain tighter Lower Bounds for this problem.

One of the other applications of adjacent BQPs is studied in Rostami et al. [12]. The authors apply a particular reformulation linearization technique to the asymmetric quadratic

traveling salesman problem and compute LBs based on Lagrangian relaxation. This problem is an adjacent BQP since only the adjacent arcs interact with each other. Moreover, they provide a cycle reformulation for the problem resulting in obtaining an exponential number of variables. Thus, they use a Column Generation approach to solve the reformulation.

Adjacent-only Quadratic Minimum Spanning Tree Problem

The QMSTP asks for a spanning tree that minimizes the objective function in (1.1). The QMSTP with adjacent edges is first studied by Assad and Xu [16]. They demonstrate that this problem is NP-hard and provide a Lower Bound based on the Gilmore-Lawyer procedure. This problem can be found in modeling some applications including communication and transportation networks. Pereira et al. [20] shows that this bound is weak and proposes a reformulation of this problem allowing a much stronger linear programming bound. They employ the concept of stars in graph theory to reformulate this problem, which is equivalent to applying a Dantzig-Wolfe (DW) reformulation of the space of stars of the graph. Since the new reformulation involves an exponential number of variables, they proceed with the obtained linear model by Column Generation. To solve the pricing sub-problems, a branch-and-cut algorithm was provided for the linearized sub-problems. In another work, they provide a branch-and-cut-and-price for the previous reformulated model of the AQMSTP and a branch-and-cut algorithm for the projection of the variables related to stars and generating exponential many new constraints [29]. Three groups of inequalities defining facets are investigated in Pereira and da Cunha [30] by analyzing the projection of the formulation on the canonical linearization variable space. Additionally, they propose four new branch-and-bound algorithms: three of them are branch-and-cut based on the projected model with inequalities, while the fourth employs a Lagrangian relaxation scheme designed for star-reformulation. Recently in Pereira and da Cunha [18], they suggest a dynamic DW reformulation that is not restricted to stars anymore. This implicit reformulation exploits different structures simultaneously, by half-space representation. After projecting out the variables assigned to the subgraphs used in the implicit DW reformulations, the algorithm separates the obtained projection cuts. Furthermore, they use a Lagrangian relax-and-cut algorithm to approximate strong bounds for the reformulation while they embed the method in a branch-and-bound algorithm.

2.2 BQP Exact Solution Methods

The main difficulty in solving BQP stems from the quadratic structure of the objective function rather than the combinatorial nature of the problem. For example, when $X =$

$\{0,1\}^m$, problem (1.1) is equivalent to unconstrained quadratic binary optimization and hence to the max-cut problem, which is, in turn, NP-hard [31].

In general, BQP problems are NP-hard, even if the linear optimization problem over the same feasible set is tractable. This is the case for many BQP problems, including the Quadratic Minimum Spanning Tree Problem [16] and the Quadratic Assignment Problem [9], while their linear counterparts can be solved in a polynomial time.

2.2.1 Linearization Techniques

One of the most natural ways to solve BQP problems with an exact method is to linearize the quadratic terms of the model and solve the resulting MILP using state-of-the-art solvers. One of the first and most applicable types of linearization is Standard Linearization Technique (SLT). However, two main concerns appear when dealing with the MILP reformulation: the increased size of the problem (in terms of variables and constraints), and the quality of the obtained dual bounds. There have been many attempts to deal with these concerns in the literature. Adams and Sherali [32], Adams and Forrester [33], and Sherali and Smith [34] provide different reduced-size MILP reformulations of the BQP, while Liberti [35] introduces a compact linearization approach for a general class of binary quadratic problems subject to assignment constraints. This approach is then revised in Mallach [36] by proposing two new necessary and sufficient conditions to achieve consistent linearization for this class of problems. Furthermore, Hahn et al. [37], Sherali and Adams [38] and Rostami and Malucelli [17] develop the RLT, which generally provides stronger MILP reformulations. The RLT is a successful and alternative method for linearizing BQPs. It generates a hierarchy of polyhedral representations, consisting of multiple levels, to handle linear and polynomial 0-1 programming problems. Each level in this hierarchy provides an explicit algebraic description of the convex hull of feasible solutions. The level of the hierarchy corresponds directly to the degree of the polynomial terms generated during the reformulation stage. During the reformulation phase, RLT constructs various polynomial factors of degree d , where d can take values from 1 to n . These factors are obtained by multiplying d binary variables x_e or their complements $(1 - x_e)$. The RLT involves two essential steps: (i) a reformulation step that generates nonlinear valid inequalities by combining constraints from the original problem, and (ii) a linearization step that replaces each product term with a single continuous variable [39, 40]. We show some examples of these types of linearizations in Appendix A.

2.2.2 Decomposition Techniques

Another relevant approach to obtain a stronger reformulation for the BQP is to use decomposition techniques. Variants of decompositions such as Lagrangian decomposition, graph partitioning, and CG methods are employed to explore the bounds of unconstrained BQP problems [41, 42]. For constrained BQP problems, Chen et al. [43] represent bounds for the BQP using a Lagrangian-based heuristic method. Some examples of using decompositions to tackle the problem of investigating bounds for some specific BQP problems are observed for quadratic knapsack problem and QMSTP [44, 18].

Moreover, there are a few papers in the literature reformulating a BQP model for a specific application into a MILP with an exponential number of variables, which is solved by CG. Aloise et al. [24] reformulate the mixed 0-1 quadratic programming model of the modularity maximization problem and solve the reformulation using a stabilized CG. In a related study, a CG heuristic is used in a districting problem to produce the best territories for the purpose of financial product pricing [45]. Rostami et al. [12] propose a lower bounding procedure for the asymmetric quadratic TSP. They reformulate the problem as a MILP with an exponential number of cycles as variables and solve the relaxation using a CG. Recently, Yarkony et al. [25] developed an extended MILP formulation for correlation clustering. They consider solving correlation clustering for several computer vision applications through CG, Benders decomposition, and Dynamic Programming (DP). Another application that has been studied through Column Generation is the AQMSTP. Pereira et al. [29] suggest a reformulation of the AQMSTP and employ row and column generation to evaluate the LBs.

2.2.3 Other Approaches

Semi-Definite Programming (SDP), quadratic reformulation, and cutting-plane methods are alternative approaches used to generate strong relaxations of BQP. In SDP, which is considered an extension of MILP reformulations, non-negativity constraints are replaced by positive semi-definiteness constraints [46, 47]. In quadratic reformulations, one must alter the objective function of a BQP problem and transform it into an equivalent convex/non-convex BQP problem to generate tighter dual bounds [48, 49]. The use of valid inequalities, which are generated and added in a cutting-plane fashion, is another approach commonly adopted in the literature [11].

2.3 Data-driven Methods in Optimization

The use of data-driven techniques for solving large-scale optimization problems has gained significant attention from researchers in recent years. These approaches provide an alternative method for tackling real-world optimization problems by identifying patterns and structures within the data to facilitate the search for solutions. According to the description presented in Chapter 1, in this section, we review some of the studies which utilize data-driven methods in optimization.

2.3.1 Structure-level

Although Mixed Integer Non-Linear Programming (MINLP) has yet to achieve the level of maturity that MILP has succeeded, the MILP community benefits greatly from pre-processing the models and data and detecting special structures of the models. One of the structures explored in the MILP literature is the single Bordered Block Diagonal (BBD) structure. Exploiting this structure in the constraint matrix of a MILP leads to Dantzig-Wolfe decomposition, Lagrangian relaxation, and branch-and-price [50, 51]. Bergner et al. [50] provide a computational proof-of-concept to show that the DW reformulation can be automated and applied to all MILPs by exploiting and rearranging the structure of the constraint matrix in various ways. They suggest a score to measure the quality of each decomposition and identify the most useful one for the DW reformulation of a MILP.

To our knowledge, there are a few methodological studies concerning structures in BQP problems. Bettiol et al. [52] tackle BQP from the CG perspective and construct an approach to study the structure of block-decomposable problems in BQP. They present two types of relaxations that acquire strong Lower Bounds for general BQP and block-decomposable BQPs specifically. The relaxations are based on DW reformulation, while CG is used as their solution method. In addition, some studies explore the linearizability of the quadratic cost matrices. Punnen et al. [13] investigate the structure of quadratic cost matrices to propose necessary and sufficient conditions for the linearizability of the quadratic Traveling Salesman Problem. In another related work, Hu and Sotirov [53] presents a linearization-based lower bounding scheme applicable to several BQP problems using a certificate for a quadratic function to be non-negative on the feasible set.

According to the aforementioned studies in the literature, we observe the following points:

- There are only a few methodical works in the literature dedicated to exploiting the structure of the BQP or a class of BQP problems to tackle solving these problems more efficiently.

- To the best of our knowledge, there is no study in the literature that investigates the adjacent BQP class of problems.
- The star-reformulation was introduced and used only for AQMSTP in the literature.

2.3.2 Instance-level

At this level, we mostly look into the literature of the studies in which some data such as optimal solutions or feasible and infeasible solutions are investigated to find patterns helping to solve a new instance of the problem.

2.3.2.1 Combining Machine Learning and Optimization

Researchers have shown increasing interest in integrating machine learning techniques with optimization problems in recent years. Bengio et al. [54] and Nguyen et al. [55] have comprehensively explored different methods wherein machine learning helps optimization models to attain more efficient solutions.

On one side, there exist many studies leveraging machine learning methods directly to solve CO problems. Standalone heuristics that do not incorporate a component such as a linear, integer, or Constraint Programming (CP) solver are developed in order to assist machine learning methods to satisfy the constraints of a Combinatorial Optimization problem. One of the first models to directly solve a Combinatorial Optimization is developed by Vinyals et al. [56], which is a sequence-to-sequence learning method, "pointer network", to find a tour for the TSP. Using a neural network architecture [57] containing an Long-Short Term Memory (LSTM) [58] and attention mechanism, they generate a new feasible solution. They, however, opt for supervised learning, which in turn, requires solving a large number of TSP instances to generate "labels" for the training data. Machine learning and reinforcement learning methods are used to solve different problems such as patient appointment scheduling [59] and routing problem of a radiotherapy device [60]. In addition, several papers have been cited in Junior Mele et al. [61] attempting to solve TSP and its variants with machine learning.

Employing data-driven approaches to deal with the uncertainty in stochastic and robust optimization problems is an additional stream of research in this area [62, 63]. Conventionally, decision-making under uncertainty assumes perfect information of the underlying information. As such, exact values for parameters and specific probability distributions of the random variables, for instance, are known in advance. In practice, however, these accurate values are simply not available, and following these traditional solution methods with imperfect information results in inaccurate and even infeasible solutions. Unlike the Robust Optimization

methods in which the random variables are modeled as uncertain variables and the decisions are devised to account for the worst case within the uncertainty set, in data-driven optimization, observations of the random variables are direct inputs of the mathematical programming models [64].

Note that, although we investigate a problem with a type of uncertainty that is due to the lack of knowledge about the constraints in Chapter 5 of this dissertation, the problems considered in that part are not stochastic problems (i.e., with stochastic parameters). Therefore, in the rest of the literature review, we mainly describe relevant studies that examine constraint customization techniques in deterministic problems.

Predictive and prescriptive models are two phases of the business analytic field. The former is mostly about analyzing the past and predicting the future, while the latter is associated with examining the most advantageous course of action that should be taken. The availability of extensive data in modern industries has led to significant advances in business analytic techniques. One of these developments is corresponding to combining prescriptive and predictive methods to employ data for prescribing optimal decisions [65, 66]. In some existing studies on the integration of prescriptive and predictive, the predictive analytic is implemented beforehand and their results are fixed. For example, the parameters of optimization models are obtained from a machine learning method and are fixed in the optimization model. Ferreira et al. [67] apply regression trees to predict demands for initial exposure of the products which have no historical sales data for an online retailer, then using these predicted demands they optimize the price of each item to maximize the revenue. This paradigm which is called predict-then-optimize in the first step maximizes the prediction power of the model and in the second step optimizes the decisions based on the predictions and associated costs [68].

However, in the vast majority of recent studies, the integration of prediction and prescription analytics is occurring in a smarter way through leveraging the optimization problem in the training step. Therefore, considering the cost of decision-making in the learning loss function, the decisions are mostly more cost-effective [69].

2.3.2.2 Embedding Predictive Models in Optimization Models

This subsection discusses the literature which considers the cases where predictive models are embedded in optimization models. We thereby provide a brief overview of some of the most related works in the following. All these studies are examples of considering predictive models as constraints of an optimization problem.

Motivated by delivery operations and data of a food service provider, Liu et al. [70] introduces

a framework to integrate travel-time prediction and order-assignment optimization to capture the behaviors of driver’s routing decision-making and include it in the optimization problem. They employ linear regression and random forest as two compatible prediction models with optimization. They further reformulate the problem to be efficiently solved by the branch-and-price algorithm. Moreover, they discuss the robust integrated model of the problem and provide two heuristics to solve the multi-period setting of the problem.

Another relevant study is a framework called JANOS [71] which allows users to embed their predictive models and their features in MILPs. JANOS currently supports neural networks, logistic regression, and linear regression. Similar to our work, it consists of two types of variables; regular and predicted. The prediction variables in this study normally define a meaningful part of the objective function as well as constraints. As opposed to our work, they do not look for hidden side constraints for a traditional optimization problem, rather they use different features and predictive variables to define a new practical problem.

The methodology introduced in Lombardi et al. [72], called Empirical Model Learning (EML) is a way of obtaining components of a complex prescriptive method using machine learning by learning relations between decidable and observable. In this conceptual model, they generate data by simulation and propose embedding a trained neural network and decision tree into a combinatorial problem like MINLP, CP, and local search. Although they also provide an embedding for linear regression in MILP, they do not present any result on this matter. Moreover, for decision trees, they limit the embedding to local search, CP, and SAT.

The idea of providing a data-driven framework to extract business rules from historical data and integrating them into General Optimization Problem (GOP) is proposed in Hewitt and Frejinger [26]. The first assumption of their framework is that the company already knows the original optimization problem which has to be solved, meaning that the GOP is given beforehand. Access to the historical data of solving this problem and the executed plans is another essential assumption in applying this approach. Therefore, the framework is suitable for situations in which current planning already exists in the company and has been executed for a period of time. The methodology of this study is also considered as integrating predictive and prescriptive analytics; however, we distinguish this work from the aforementioned studies which attempt to estimate the parameters of the optimization models or those replacing an optimization approach with a machine learning approach. More specifically, in Hewitt and Frejinger [26], they present a framework for embedding linear regression models which are trained from historical implemented solutions as constraints to adjust the values of the decision variables in (deterministic) optimization models.

2.3.2.3 Constraint Acquisition

It is noteworthy to mention that in the study of Chapter 5, the optimization model is known, and the only missing part is one or a small set of constraints. We use statistical learning to learn these hidden constraints. Therefore, we do not review studies related to the terms such as model learning or model seeking [73, 74].

The third study of this dissertation in Chapter 5 should be compared with CP, and constraint acquisition. Studies in constraint acquisition focus on learning a set of constraints based on a bunch of feasible and infeasible examples. The vast majority of these approaches attempt to match the examples with a combination of rules in a given library and suggest constraints that are compatible with the solutions in hand [75, 76]. Although most of the constraint acquisition methods synthesize constraints for CP models, a few studies such as Pawlak and Krawiec [77] and Schede et al. [78] learn constraints for MILP or Non-Linear Programming (NLP) models. A comprehensive survey of related studies is represented in Fajemisin et al. [79]. What differentiates their work from the literature in integrating machine learning and optimization is that these approaches do not take advantage of the power of statistical learning and predictive models to learn complex decision boundaries.

2.3.2.4 Inverse Optimization

Inverse optimization is another research field using data-driven approaches to infer optimization model parameters based on observed data points of optimal solutions. This area of research originated from an investigation into the inverse shortest path problem for the first time [80]. In data-driven inverse optimization, the goal is to minimize particular loss functions that penalize the extent to which inverse feasibility is not met [81]. Ahmadi et al. [82] propose a data-driven inverse linear optimization framework to infer optimal solutions and unknown parameters directly based on observations (feasible and infeasible solutions) and known parameters.

The majority of the literature considers determining parameters of the objective function of the problems where they assume the feasibility region is known. There are only a few studies considering the recovery of constraint parameters. In Chan and Kaw [83], two approaches are suggested to impute unknown left-hand side coefficients of constraints as well as cost vector of objective functions for linear optimization problems. Their methods are based on minimizing the duality gap and satisfying strong duality in order to make an observed solution minimally sub-optimal with respect to some cost vectors. However, in our study in Chapter 5, we only learn side constraints that are not part of the original optimization model

in a separate learning step. Hence, we embed them in the optimization model.

Based on the reviewed studies, we obtain the following observations:

- The majority of companies modify the solutions derived from optimization-based decision support systems in accordance with the internal business conventions of the company or the experts' preferences.
- To tailor the optimization models to the specific requirements of individual companies, it is essential to conduct interviews with the company's experts and managers. In addition, it may be difficult to formulate the logic behind the preferences. Hence, this process hugely increases the time and costs associated with the development of the decision support system.
- To reduce the corresponding cost, it is critical to enable the utilization of unified software across multiple companies and automate the customization process.
- In the situation that historical data is available to represent the optimal solutions obtained by classical models used in the past and the plans implemented in the company, the integration of predictive models with prescriptive models through a data-driven framework is promising.
- There are only a few studies investigating such a data-driven algorithm to customize the constraints of classical optimization models.
- To the best of our knowledge, there exists only one study that proposes and applies a data-driven approach to address this problem, which can be efficiently solved using MILP [26].
- In Hewitt and Frejinger [26], a linear regression method is used to learn the hidden business rules.

CHAPTER 3 STAR-REFORMULATION FOR ADJACENT-ONLY BINARY QUADRATIC PROGRAMMING PROBLEMS

3.1 Introduction

BQP is a class of Combinatorial Optimization problems comprising binary variables, quadratic objective functions, and linear/non-linear constraints. In general, BQP is known to be NP-hard, even if the linear optimization problem with the same set of feasible solutions is computationally manageable.

As we mentioned in Chapter 1, researchers have been increasingly interested in efficient data-driven methods recognizing inherent patterns and structures within large-scale optimization problems to tackle real-life problems. Some examples in this field which are more related to the study of this chapter include investigating the structure of constraint matrices [8] or the quadratic cost matrices [13] and identifying significant patterns within specific problem classes [6].

The most related works to our study are the papers devoted to the reformulation of the Adjacent-only Quadratic Minimum Spanning Tree Problem to a Mixed Integer Linear Programming and solution algorithms applied to solve the reformulation. More specifically, a star-reformulation is introduced in Pereira et al. [20] based on the concept of stars in graph theory to reformulate the AQMSTP by a stronger LP. Then, an algorithm based on dynamic column and row generation is proposed to determine the dual bounds of the problem. In another study, Pereira et al. [29] presents a branch-and-cut-and-price algorithm based on this reformulation of the AQMSTP and a branch-and-cut algorithm based on projecting out the decision variables of this model. There are also other reformulations and algorithms for the AQMSTP in the literature based on the proposed star-based model (e.g., Pereira and da Cunha [30, 18]).

To the best of our knowledge, the idea of exploring the effectiveness of the star-reformulation on more general cases rather than the AQMSTP has not been investigated in the literature. Our major focus here is to explore the generalizability and usability of this reformulation on other problems in the adjacent-only class of the BQP problems.

3.1.1 Main Contributions

The main contributions in this chapter¹ are summarized as follows:

- We investigate the effectiveness of the idea of star-reformulation proposed for the AQM-STP in the literature on the class of adjacent-only BQP problems.
- We develop CG approaches for each specific application to derive valid dual bounds.
- To demonstrate the potential of the star reformulation on other problems in adjacent BQP classes, we consider the AQSAP, and MOT as two different adjacent BQP problems whose reformulations lead to different master problems and pricing subproblems.
- We perform extensive computational experiments to evaluate the proposed reformulations and CG algorithms on the adjacent-only problems. We depict that the special quadratic matrix structure of adjacent-only BQPs derives the most significant benefit from the star-reformulation and CG.

The rest of this chapter is organized as follows. In Section 3.2, we introduce the adjacent-only BQP class of problems and present the idea of star-reformulation. Section 3.3 is corresponding to a Column Generation method to solve the reformulation. Section 3.4 is dedicated to presenting and reformulating two adjacent-only BQP problems, the AQSAP and MOT, as our illustrative examples to generalize star-reformulation. Finally, in Section 3.5, we explain our test sets and display the computational results from the introduced problems, and in Section 3.6, we wrap up the chapter and provide some future possible directions for the research.

3.2 Star-Reformulation For Adjacent-only BQP Problems

As we discussed in the chapters related to the introduction and literature review, in numerous applications of BQP models, the interaction costs exclusively exist between adjacent edges. Consequently, pairs of edges that do not have a shared endpoint have zero quadratic costs. In Pereira et al. [20], an idea of star-reformulation is introduced to reformulate and efficiently solve the Adjacent-only Quadratic Minimum Spanning Tree Problem. To exploit such a unique structure of the adjacent BQP problems in developing a solution strategy, we follow Pereira et al. [20] to reformulate the general class of adjacent-only BQPs to a new model

¹This chapter is based on the following article: Bayani, M., Rostami, B., Adulyasak, Y., Rousseau, LM. A Dual Bounding Framework for Binary Quadratic Combinatorial Optimization, Les Cahiers du GERAD ISSN. 2021 Aug;711:2440. Submitted to INFORMS Journal on Computing in July 2021 (Under 3rd revision).

with exponentially many variables based on the concept of stars in a graph. To this end, we first provide some notations to introduce the concept in Section 3.2.1 and then present the star-based reformulation in Section 3.2.2.

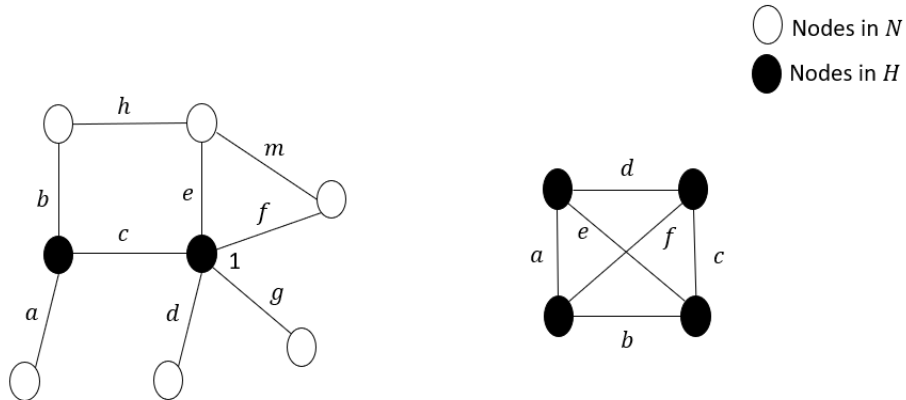
3.2.1 Definitions

Consider the BQP problem on graph $G = (V, E)$ in (1.1). Without loss of generality, we assume that $V = N \cup H$ where N and H are disjoint sets. We also take into account the possibility of N being an empty set. Thus, in this situation, the set V is equal to the set H . For each $v \in H$, we define $\delta(v) \subseteq E$ as the set of edges incident to node v and let $A = \cup_{v \in H} \delta(v)$ be the set of all edges with exactly one endpoint in H , considering exceptionally, when the set N is empty, both endpoints of all edges of the related graph are thereby included in H . So, if $N \neq \emptyset$, A is the set of edges with exactly one endpoint in H , and if $N = \emptyset$, then $A = E$. Two distinct edges of A , say $e = \{i, j\}$ and $f = \{k, \ell\}$, are adjacent if they share a common endpoint v in H , i.e., if $\{i, j\} \cap \{k, \ell\} = v \in H$. We denote by $\mathcal{A}$ the set of distinct pairs of adjacent edges in A :

$$\mathcal{A} = \left\{ (e, f) \in A \times A : e = \{i, j\}, f = \{k, \ell\}, \{i, j\} \cap \{k, \ell\} = v \in H \right\} \quad (3.1)$$

The graphs indicated in Figure 3.1 illustrate the concept of adjacent edges by defining sets A and $\mathcal{A}$.

Figure 3.1 Two examples of graph to demonstrate the concepts of set A , adjacency set $\mathcal{A}$ and star s



Note: Graph on the left: $A = \{a, b, d, e, f, g\}$, $\mathcal{A} = \{(a, b), (d, e), (d, f), (d, g), (e, f), (e, g), (f, g)\}$

Graph on the right: $A = \{a, b, c, d, e, f\}$,

$\mathcal{A} = \{(a, b), (a, d), (a, e), (a, f), (b, c), (b, e), (b, f), (c, d), (c, e), (c, f), (d, e), (d, f)\}$,

Node 1 is specified to motivate a star center candidate

These definitions will lead to the following definition for the star-shaped sub-graph s . That is, for each $v \in H$ we define a star s centered at node v as any subset of $\delta(v)$ whose elements are in the set A , including an empty set and let S^v be the set of all stars centered at node v . The mathematical representation of the aforementioned definitions can be expressed as:

$$S^v = \{s : s \subseteq (\delta(v) \cap A)\}.$$

Therefore, $S = \bigcup_{v \in H} S^v$ includes all the possible stars centered at nodes $v \in H$ in the graph. As an example, in the left graph of Figure 3.1, $s = \{e, f, g, d\}$ is the largest possible star centered at node 1. Note that as both endpoints of the edge, c are in the set H , this edge is not included in any possible star of this graph.

3.2.2 Star-Reformulation

In the adjacent-only cases, interaction costs are zero for pairs of edges that do not share a common endpoint in the defined subset H . Due to equation (3.1), in this BQP class, the quadratic cost q for the pairs of edges (e, f) that are not covered by the set $\mathcal{A}$ are zero. Based on the represented concepts and the fact that each star $s \in S$ consists of a subset of pairs in $\mathcal{A}$, we can reformulate an adjacent-only BQP problem in terms of stars. For each $s \in S$ let $C_s = \sum_{e \in s} c_e + \sum_{e, f \in s} q_{ef}$ represent the total linear and quadratic cost of star s . We define a new binary decision variable λ_s for each star $s \in S$ to indicate if the corresponding star s is included in the solution of the BQP problem or not. We also denote $\mathcal{F}$ to depict the feasible space of the problem. By preserving the definition of variable $\mathbf{x} \in X$, we provide the following star-based reformulation:

$$\min \quad \sum_{s \in S} C_s \lambda_s + \sum_{e \in E \setminus A} c_e x_e \quad (3.2)$$

$$\text{s.t.} \quad (\mathbf{x}, \boldsymbol{\lambda}) \in \mathcal{F}(\mathbf{x}, \boldsymbol{\lambda}) \quad (3.3)$$

$$\mathbf{x} \in \{0, 1\}^m \quad (3.4)$$

$$\boldsymbol{\lambda} \in \{0, 1\}^{|S|} \quad (3.5)$$

The objective function of the reformulation minimizes the total cost of the problem and consists of two different parts. The first part corresponds to the cost of the star s , including the linear costs of the edges inside the star and the interaction between adjacent edges of that star. The second term of the objective function reflects the linear cost of the edges which are not incorporated in any possible star. Constraint (3.3) links the feasible region of

the problem to the stars by coupling the original variables $\mathbf{x}$ and new variables $\boldsymbol{\lambda}$. It can also include the constraints which are only related to the variables $\boldsymbol{\lambda}$ and the constraints which are only associated with variables $\mathbf{x}$. We assume, without loss of generality, that such linking constraints can always be found because, for each $v \in H$ and each $e \in \delta(v)$, there exist parameters $b_{es} \in [0, 1]$ such that $x_e = \sum_{s \in S^v} b_{es} \lambda_s$ (e.g., see Section 3.4). Since some of the constraints in the original model can be included in the definition of the star in this reformulated counterpart, the $\mathbf{x}$ -only constraints can be considered as a subset of X in (1.1).

To elaborate on the star reformulation consider the right graph of Figure 3.1. In this graph, a given set of star centers (H) potentially comprises the whole node set of the graph (V) (i.e., $H = V$). Intuitively, for the BQP problems with this property, set A contains all the graph's edges. To illustrate this, consider the AQMST problem [16], in which the set H is equal to all nodes of the graph ($N = \emptyset$). Thus, in the reformulated model of the AQMST of Pereira et al. [20], the objective function consists of only a linear term to represent the cost of stars. Nonetheless, in the left graph of Figure 3.1, some of the nodes are not considered as possible star centers ($N \neq \emptyset$), so we keep the second term of the reformulation's objective function to demonstrate the linear cost of the edges which are not included in any possible star. In this study, we employ the star reformulation to model two applications, more details on modeling and solution methodology are presented in Section 3.4.

3.3 Column Generation Approach

The use of the star representation in the proposed reformulation of the adjacent BQP in (3.2)-(3.5) results in an exponential number of variables $\lambda_s, s \in S$.

Pereira et al. [20] suggest a Column Generation approach for star-reformulation of the AQMST problem. Inspired by this idea, CG is applied to deal with the general star reformulation of (3.2)-(3.5).

CG is an efficient iterative algorithm for providing dual bounds for problems with an exponential number of variables (columns) [84]. In each iteration of the algorithm, it solves one restricted master problem (RMP) which is the problem restricted to a small subset of the variables, and one or several pricing subproblems. Using the dual information of the RMP, the pricing subproblem is solved to verify the optimality of the master problem and the CG algorithm stops if the optimality condition is satisfied. Otherwise, one or more new variables determined by the subproblem will be added to the RMP and the updated RMP will be solved in the new iteration.

Consider $\bar{S} \subseteq S$ as a subset of possible stars in the star-reformulation proposed in Section

3.2.2, hence we obtain the following RMP for the LP relaxation of the problem:

$$\min \quad \sum_{s \in \bar{S}} C_s \lambda_s + \sum_{e \in E \setminus A} c_e x_e \quad (3.6)$$

$$\text{s.t.} \quad (\mathbf{x}, \boldsymbol{\lambda}) \in \bar{\mathcal{F}}(\mathbf{x}, \boldsymbol{\lambda}) \quad (3.7)$$

$$\mathbf{x} \in \{0, 1\}^m \quad (3.8)$$

$$\boldsymbol{\lambda} \in [0, 1]^{|\bar{S}|} \quad (3.9)$$

where $\bar{\mathcal{F}}(\mathbf{x}, \boldsymbol{\lambda})$ is a subset of $\mathcal{F}(\mathbf{x}, \boldsymbol{\lambda})$ restricted to $\bar{S}$.

In each iteration, we add a subset of columns $s \in S \setminus \bar{S}$ that potentially improves the objective function of (3.6). To this end, we solve an auxiliary problem to find the most negative reduced cost column to add to the master problem. Thus, a column entering the basis can be found by computing the minimum reduced cost star with respect to the quadratic and linear costs of the edges inside the star.

According to the definition of the cost of stars C_s , the pricing subproblem can either be a linear problem, or a BQP problem. In a simple setting, this subproblem can take the form of a Unconstrained Binary Quadratic Programming (UBQP). Nevertheless, it is possible that one must explicitly incorporate constraints in the binary quadratic subproblem in some specific applications. Given that the essence of solving a BQP problem exactly is NP-hard, intuitively adding columns with negative reduced cost without solving the subproblem to optimality can be a promising alternative when applying the CG algorithm. However, to provide a/some valid dual bound(s), we need to solve the subproblem to optimality [24, 45]. Specifically, in the case of a UBQP subproblem, several solution approaches based on greedy and heuristic methods are proposed to solve this problem [85]. Even in the case where the pricing is a constrained BQP problem and obtaining an exact solution is necessary, the size of the problem is much smaller than the compact formulation in (1.1), meaning the problem is relatively easy to solve. In addition, when the subproblem is a constrained BQP problem, we have better options in terms of linearization techniques such as RLT to solve it exactly. In the following section, we use examples to demonstrate how to deal with each of these cases.

As we observe in the proposed reformulation, special structure assumptions for the quadratic matrix $\mathbf{q}$ to have only adjacent quadratic costs offer promising properties for solving BQP problems. In this reformulation, the stars interact with each other only through linear costs. The new formulation for this particular class is an integer linear problem which is more tractable to solve than the generic model of Section 1.1. Noteworthy mentioning that the quadratic interaction between two adjacent edges may depend on their common endpoint, or

may be independent of it, resulting in different sparsity levels of the matrix $\mathbf{q}$. We investigate this concept in more detail through our examples in the next sections.

3.4 Illustrative Examples

The objective of this section is to demonstrate how the star-reformulation and the solution strategy described in Sections 3.2 and 3.3 can be applied to different adjacent-only BQP problems. To this end, we consider two BQP problems whose reformulations lead to master problems and pricing subproblems, described as follows:

- Linear programming master problem and unconstrained BQP pricing subproblem
- Linear programming master problem and constrained BQP pricing subproblem

We consider the Adjacent-only Quadratic Semi-Assignment Problem and the Multiple Object Tracking problem, to outline the computational advantages of star reformulation in this class. In the following subsections, we provide a brief description and literature review for each problem, as well as the compact BQP models. Then, we describe how to reformulate them as (3.2)-(3.5) and obtain dual bounds using our CG solution method.

3.4.1 Adjacent-only Quadratic Semi-Assignment Problem

In the QSAP, we are given a set of clients $N = \{1, \dots, n\}$ and a set of servers $H = \{1, \dots, h\}$. In this problem, there is a linear cost $c_e, e = \{i, j\}$, associated with the assignment of client $i \in N$ to server $j \in H$ and there is a quadratic cost $q_{ef}, e = \{i, j\}, f = \{k, l\}$, associated with the assignment of client $i \in N$ to server $j \in H$ and client $k \in N$ to server $l \in H$ simultaneously.

In this section, we consider a special class of the QSAP in which the quadratic costs are only restricted to the adjacent edges. One notable application is the QSAP of assigning n clients to h servers, this time in a distributed processing system. If client $i \in N$ is assigned to server $j \in H$, the required processing time c_{ij} , is computed based on the processing speed of the server and the client's demand. In this type of problem, where multiple clients are assigned to the same server, there is no predefined priority and the order of processing the requirements is unknown. In this situation, every client $i \in N$ aims to minimize the worst-case completion time. Hence, the completion time of client i is set to $CT_i = \sum_{j \in H} x_{ij} \sum_{k \in N} c_{kj} x_{kj}$, where the binary variable $x_{ij} = 1$ indicates assigning client i to server j [86]. The goal in this problem is obtaining an assignment to minimize the total completion time for all clients, $\sum_{i \in N} CT_i$.

If we transform this problem to the previously defined graph G with node set V and edge set E , according to the definitions presented in Section 3.2.1, the compact model of AQSAP is:

$$\min \quad \sum_{e \in A} c_e x_e + \sum_{(e,f) \in \mathcal{A}} q_{ef} x_e x_f \quad (3.10)$$

$$\text{s.t.} \quad \sum_{e \in \delta(i)} x_e = 1 \quad \forall i \in N \quad (3.11)$$

$$x_e \in \{0, 1\} \quad \forall e \in E. \quad (3.12)$$

Remark that for the specific explained problem, the equation below holds:

$$q_{ef} = c_e + c_f. \quad \forall (e, f) \in \mathcal{A} \quad (3.13)$$

This formulation is valid for all assignment problems in which multiple clients compete for a single machine and each assigned client has to undergo the completion time of the machine. This problem includes a large class of the QSAP, while the structure of its quadratic matrix narrows it down to the class of problems described in Section 3.2. Note that the explanations provided here are based on an application of AQSAP for clarity. Nevertheless, AQSAP is a general problem and can be applied in various applications.

3.4.1.1 Reformulation.

We observe two fundamental properties of the current problem: (i) The linear costs for the edges that are not covered by stars are zero, and (ii) the non-adjacent edges do not interact with each other. Therefore, the second property in turn leads to the star-reformulation of (3.2)-(3.5).

To this end, without loss of generality, we assume that every server j is a center of a star-shaped sub-graph s . So, the binary variable λ_s corresponds to selecting this star. We define parameter $B_{js} \in \{0, 1\}$ to indicate if server j is the center of star s or not, and the parameters $D_{is} \in \{0, 1\}$ to identify if client i is included in star s . So according to the notations given

above, the star reformulation is as follows:

$$\min \quad \sum_{s \in S} C_s \lambda_s \quad (3.14)$$

$$\text{s.t.} \quad \sum_{s \in S} B_{js} \lambda_s \leq 1 \quad \forall j \in H \quad (3.15)$$

$$\sum_{s \in S} D_{is} \lambda_s = 1 \quad \forall i \in N \quad (3.16)$$

$$\lambda \in [0, 1]^{|S|}, \quad (3.17)$$

where constraints (3.15) impose that at most one star can be chosen among all the stars centered at j . Constraints (3.16) are the set partitioning linking constraints, enforcing that each client must be included in exactly one star.

3.4.1.2 Column Generation.

In this case, the reformulation is linear and CG is directly applicable. In order to solve the reformulation with CG, we start from a feasible subset of stars and solve the RMP of the star reformulation in (3.14)-(3.17). Then the pricing subproblem looks for the star with the minimum reduced cost for each server. This pricing can be written on graphs as follows for every star center j :

$$\min \quad \sum_{e \in \delta(j)} (c_e - \rho_e) z_e + \sum_{e, f \in \delta(j)} q_{ef} z_e z_f - \pi_j \quad (3.18)$$

$$\text{s.t.} \quad z_e \in \{0, 1\} \quad \forall e \in \delta(j) \quad (3.19)$$

where $\pi_j, j \in H$, and $\rho_i, i \in N$ are the optimal duals associated with constraints (3.15), and (3.16), respectively. Since ρ_i is fixed for each i and is not dependant on j , we can transfer ρ_i on the edges incident to j to have a more consistent formulation using ρ_e , where $e = \{i, j\}$. The decision variable of the problem is presented using $z_e, e = \{i, j\}$, which is equal to 1 if client i is assigned to star center j and 0 otherwise. Therefore, we end up with several UBQP pricing subproblems. Implementation details of the CG approach, and computational experiments to find dual bounds, are provided in Section 3.5.

3.4.2 Multiple Object Tracking

Multiple object tracking (MOT) and, more specifically, multiple people tracking is a well-known application in computer vision that aims to track multiple objects (people) in a sequence of video frames. MOT is associated with a variety of applications like self-driving

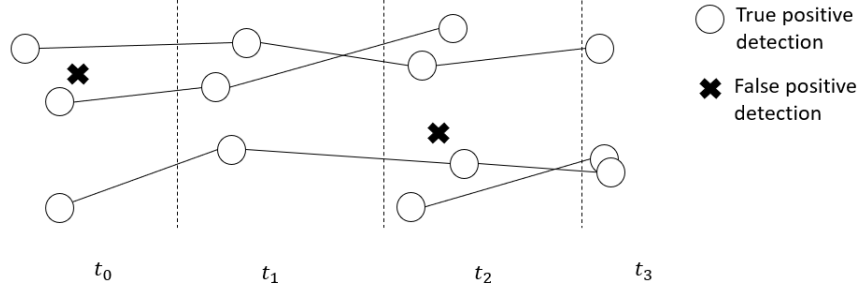
cars, human-computer interaction, security and video surveillance, sports analysis, some games like Microsoft Kinect, traffic analysis, etc. [87, 88]. Despite recent developments in this area, it is still a very challenging task due to occlusion and scene cluttering. As a result of the advancement of object detection technologies, detection-based methods are the most dominant techniques in MOT [89, 87]. MOT consists of three main components: (i) detecting the objects, in which a person detector is utilized for each individual frame to find the potential locations of all the people, (ii) affinity, or score estimation, which demonstrates how likely detections are related to a single identity, and (iii) data association, in which these hypotheses are linked across the frames based on the estimated scores to form tracks [4].

In general, while object detection and score determination are deep learning tasks, data association is a combinatorial optimization problem. Once the detections and their unary and pair-wise scores are computed, they are given as inputs to the data association problem to generate the associated tracks. In MOT, a sequence of frames (from t_0 to t_3) in a scene of a few objects (such as people) is depicted in Figure 3.2. These potential objects are identified as a priori through the use of a machine-learning-based object detection method that provides a list of potential objects and the likelihood scores associated with them. This list and the detection scores are then used as the inputs in the optimization model. The detected objects are represented by circles and cross marks in this figure where the detection scores produced by the detection algorithm are included as a part of the linear and quadratic costs in the MOT model. This figure represents the results of the data association step. In this step, an optimization model is solved to determine the set of detections that belong to the same object across multiple frames based on the cost parameters. The figure shows that, after solving the optimization model, the solution indicates four tracks (people), which are depicted by connecting lines in the figure, and two false detections, represented by cross marks. As shown in Figure 3.2, two of the objects are captured in all four frames while one of the objects moves through frames t_0 to t_2 and the other object only moves from frame t_2 to the last frame.

Data association algorithms can be categorized as online or offline algorithms [88]. Here we consider the offline data association case.

Essentially, a data association problem can be modeled with respect to a graph $G = (V, E)$. A detection $i \in N$ is represented by a node in this graph. We consider another set of nodes $H = \{1, 2, \dots, h\}$ which are dummy nodes related to the tracks (target people). h is a Upper Bound on the number of target people in the video, which is predefined as an input. More precisely, we define the graph where the vertex set V consists of all detections, i.e., potential bounding boxes of potential candidates of people N in a video sequence and all possible tracks (target people) H . Similar to Section 3.2, consider A as a subset of edges E that

Figure 3.2 Graphical representation of a solution of data association in MOT applied to detections of four video frames



are incident in a node in H . Therefore, edge $e = \{i, j\} \in A$ denotes a possible linking of detection to a track (person).

Given $T = \{1, 2, \dots, \mathcal{T}\}$ as the set of all frames, each detection i belongs to a frame $t \in T$. We introduce two other subsets of edges based on our definitions in Section 3.2.1: $\delta(i) \subseteq A$ is a subset of edges in A incident to node i and $\delta^t(i)$ is a subset of $\delta(i)$ when the edges stem from the frame t .

Each edge $e = \{i, j\} \in A$ has a cost $c_e \in \mathbb{R}$ defined via a logit function and reflects the likelihood of detection $i \in N$ being a correct detection. This cost is called unary cost in the computer vision literature [4]. Unary cost is fixed for each detection i and is not dependant on tracks. For every pair of edges (e, f) which are incident in a node in H , a pair-wise cost $q_{ef} \in \mathbb{R}^{m \times m}$ is to be paid. Note that the interaction between edges e and f is non-zero if, and only if, the detections i and k are assigned to a distinct track. Pair-wise cost identifies how likely two detections belong to the same person. The probabilities which determine the costs are inferred based on detection scores. There are several ways to estimate these score terms from the geometry features, color histogram, appearance, and other features related to the image data. Considering p as this probability, the quadratic cost is computed as $\log(1 - p)/p$, so depending on the unary and pair-wise probabilities, the costs can be negative, positive or zero, resulting in non-convexity of the problem [90, 4].

Similar to the model presented in Henschel et al. [4], the MOT data association problem to

minimize the total cost of labeling is expressed as:

$$\min \quad \sum_{e \in A} c_e x_e + \sum_{(e,f) \in A} q_{ef} x_e x_f \quad (3.20)$$

$$\text{s.t.} \quad \sum_{e \in \delta(i)} x_e \leq 1 \quad \forall i \in N \quad (3.21)$$

$$\sum_{e \in \delta^t(j)} x_e \leq 1 \quad \forall j \in H, \quad \forall t \in T \quad (3.22)$$

$$x_e \in \{0, 1\} \quad \forall e \in A, \quad (3.23)$$

where constraints (3.21) are needed to mandate that more than one track assignment is not possible for every detection i . Constraints (3.22) restrict the model to select at most one detection associated with each track inside every frame.

Although the BQP model from Henschel et al. [4] only contains constraints (3.21), different types of potential side constraints can be added to this primary model. These constraints can be formulated based on prior knowledge or hypotheses associated with the scenes that video frames come from, the types of cameras taping the frames, or the features of the object detector as well as other constraints based on the prior knowledge with respect to the objects in the scenes [91, 90]. The goal here is to input possible additional knowledge in the hand of the scenes, detections, and other available information as the constraints to the model. Thus, the obtained tracks will be more accurate with respect to the ground truth. In this study, based on the assumption of using only one detector (body detector) and having one detection per person as inputs to the problem, we append the frame constraints in (3.22) to the basic model from Henschel et al. [4], guaranteeing that no two detections inside a frame are associated with the same person.

There are a few works in the literature on object tracking which are related to our modeling and solution method. Leal-Taixe et al. [92] study the problem of tracking multiple objects across multiple cameras. Their LP minimum cost flow formulation of the problem has block structural properties and they explore the results using a branch-and-price algorithm. Wang et al. [93] model the MOT problem through an ILP and suggest using CG for MOT and solving the associated pricing subproblem with dynamic programming. They consider a pool of constructed tracklets (short tracks along with a few frames) as input for their model and choose an optimal among them based on DP. However, in this study, we deal with the quadratic interactions between many detections simultaneously.

3.4.2.1 Reformulation.

To reformulate the problem as the star-reformulation model, (3.2) to (3.5), we identify each track j as the center of a possible star s . According to the definition of the problem, the pairwise costs of MOT correspond only to the pairs of edges associated with the same person, meaning that if the edges are adjacent in H , their corresponding quadratic cost is non-zero and otherwise it is zero. Therefore, we can exploit the special structure of the adjacent-only class in this problem. Moreover, since the cost c_e is zero for the edges that are not incident to nodes in H , the objective function of our reformulation is reduced to the cost of stars. Thus, the linearized Restricted Master Problem (RMP) of the star-based reformulation of MOT is given below:

$$[\text{RMP-MOT}]: \quad \min \quad \sum_{s \in \bar{S}} C_s \lambda_s \quad (3.24)$$

$$\text{s.t.} \quad \sum_{s \in \bar{S}} \lambda_s \leq h \quad (3.25)$$

$$\sum_{s \in \bar{S}} D_{is} \lambda_s \leq 1 \quad \forall i \in N \quad (3.26)$$

$$\lambda \in [0, 1]^{|\bar{S}|} \quad (3.27)$$

This model consists of one star-only constraint (3.25) enforcing the maximum number of tracks, and one set of coupling constraints (3.26) to impose labeling every detection with at most one track.

3.4.2.2 Column Generation.

The CG process starts from an empty set of feasible stars where we solve a pricing subproblem for each person $j \in H$ as a star center. Let π and $\rho_i, i \in N$, be the optimal solutions of dual variables associated with constraints (3.25) and (3.26), respectively, and by transferring the definition of ρ_i to the edge $e = \{i, j\}$, we rewrite it as ρ_e . We further define binary variable $z_e = 1$ when edge e is selected in the star and $z_e = 0$ otherwise. Given this, the pricing subproblem corresponding to center $j \in H$ is as follows:

$$\min \quad \sum_{e \in \delta(j)} (c_e - \rho_e) z_e + \sum_{e, f \in \delta(j)} q_{ef} z_e z_f - \pi \quad (3.28)$$

$$\text{s.t.} \quad \sum_{e \in \delta^t(j)} z_e \leq 1 \quad \forall t \in T \quad (3.29)$$

$$z_e \in \{0, 1\} \quad \forall e \in \delta(j) \quad (3.30)$$

where constraint (3.29) restricts each star to select a maximum of one detection per frame.

We should remark that this CG process requires only one subproblem to be solved at each iteration. This is due to the fact that neither linear nor quadratic costs are dependent on the star centers; the centers can be realized identically. More specifically, suppose that $j, l \in H$ are possible star centers (tracks) and $i, k \in V \setminus H$ are the detections. The quadratic interaction between $e_1 = \{i, j\}$ and $f_1 = \{k, j\}$, $q_{e_1 f_1}$ is equal to the quadratic interaction between $e_2 = \{i, l\}$ and $f_2 = \{k, l\}$, $q_{e_2 f_2}$.

Note that the pricing subproblem in this case is a constrained BQP problem. The implementation details are provided in Section 3.5.2.

3.5 Computational Experiments

In this section, we provide a rigorous experimental study to evaluate our suggested framework on test instances of the adjacent-only QSAP and MOT in terms of dual bound and computing time. We attempt to answer this fundamental question through our experiments: How effective and applicable is the star-reformulation on other adjacent-only problems like AQSAP and MOT?

To answer this question, we compare star-reformulation and CG framework with both SLT and RLT as the two most commonly used methods in the literature of BQP. For the AQSAP, we also compare our results with the most effective exact method proposed [49]. GUROBI version 9.0.1 is chosen as our benchmark Mixed Integer Programming (MIP) solver and we also solve the BQP models directly using GUROBI. Note that, through the CG procedure, we consider various possible reformulations which result in different pricing subproblems (i.e., BQP and different linearizations). We consider a time limit of 3 hours to solve each instance.

For the sake of comparison, the root node dual bound of the RMP is selected as a reliable indicator to be compared with the solver dual bound. Yet, the dual bound is not the only measure of efficiency; we single out the computation time needed to complete the CG process in the root node and for GUROBI to solve the problem.

We compare the trade-off between computation time and the quality of the dual bound using the methodology of performance profiles [94, 50] in the related section of each problem. The profile plot shows the cumulative distribution function (CDF) of the ratio of the time taken by each method to solve each problem or the bound obtained by each method to the best acquired time or bound among all the approaches for that problem. These two criteria are described below in more detail:

- **Dual bound performance profile:** The first set of graphs is based on the dual bound quality regardless of the time required to compute that bound. For each instance and each method, we compute the ratio between the dual bound of each method and the best bound among them. The horizontal axis reports this ratio, thus the vertical axis corresponds to the fraction of instances with at least this ratio of bound performance displayed for each method. A large value is considered for the ratio where the method could not provide any dual bound for an instance within the time limit.

- **Time performance profile:** The second type of graph is generated based on the time needed to obtain the best dual bound. In this analysis, for each instance, we first find the best dual bound among all the obtained dual bounds by different methods. Then, for every method which yielded the best dual bound, we consider the ratio between the time required by that method to attain the best dual bound and the shortest time among all of them. This ratio provides the performance index in the horizontal axis. We report the fraction of instances with a maximum of a specific time ratio in the vertical axis. A large value is assigned to the ratio where the method is not able to achieve the best dual bound for an instance. Before moving on, let us make the following remarks:

Remark 1. For each method, the probability that the method will win over the rest of the methods is defined by the fraction of instances with the best performance [94]. Hence, we refer to the ratio of instances with the performance equal to one, as the number of wins of the associated method.

Remark 2. When running the CG algorithm, we do not have to wait until the termination of the CG procedure to obtain a valid LB. If the CG procedure could not converge in our desired time, we have information about the intermediate quality of the dual bound in each iteration at the expense of slightly more computations [95]. Nonetheless, to attain an exact dual bound, we must solve the pricing of that iteration to optimality.

Remark 3. Primal bounding methodologies are beyond the scope of the current study; thus, we do not embed the CG procedure in a branch-and-bound tree. However, we compute primal bounds by applying a trivial heuristic to the solutions of the CG approach for each instance of the problem. In this heuristic, we solve the IP model for the master problem of

the last iteration where all available columns are considered as binary variables. In Appendix A.3, we discuss the obtained Upper Bound (UB)s in more detail.

Remark 4. The reported computation time for the experiments is comprised of the time to attain both LB and UB.

Remark 5. It is well-accepted that difficulties in proving optimality may appear when column generation is solving a degenerate, large-scale problem. In addition, dual variables may oscillate from a good one to a much worse one, deriving the same value for many iterations of CG [96, 97]. Computational experiments show that it is possible to alleviate these effects using stabilization techniques such as dual-optimal inequalities and stabilized Column Generation algorithms. In this study, we implemented BoxPen, in-out separation, and interior-point stabilization techniques [98, 99, 100]. Since these methods need tuning of several parameters and our main focus is on the application of the standard CG algorithm, further specialized CG enhancements and tuning procedures for specific problem structures were not thoroughly investigated. Our computational results demonstrate improvements in some instances, but not in all of them. Thus, we have decided to keep the unstabilized results in our reports. However, one can test different techniques to accelerate the CG algorithm to solve the proposed star-reformulation in an arbitrary application.

All the algorithms and models were implemented in the Python programming language. They were performed on a shared cluster with a four-core, 3.05GHz processor and 128GB RAM running under Linux 7.8. In the following sections, we present the test instances, parameter settings, and computational results for AQSAP and data association in MOT. The instance-by-instance tables in Appendix A provide more details on the results of each problem.

3.5.1 Adjacent-only QSAP Experiments

Instances in the literature which can be used for the variants of the QSAP experiments are very limited and many papers rely on data instances that were randomly generated [101, 49]. We generate random instances for the AQSAP, introduced in Section 3.4.1, where the processing cost c_{ij} for assigning a client i to a machine j is computed as $dm_i \times pr_j$. The client unit of demand for processing i is identified by dm_i , while pr_j indicates the required time for processing a unit in machine j . They are randomly generated over $(0, 100)$ and $(0, 10)$, respectively, from uniform distribution. We carry out our experiments considering different combinations of parameters $n \leq 50$ and $h \leq 14$ since in practice h is smaller than n . In this experiment, 41 instances are generated by altering parameters n and h which are defined in the associated tables in Appendix A.

As mentioned previously, the pricing subproblem (3.18) is an unconstrained BQP problem. In order to heuristically solve this, we employ an open-source solver, qbsolv [102]. Based on divide and conquer and dynamic programming, the solver partitions the problem into multiple subproblems and solves them using a tabu search algorithm. When the heuristic solver fails to find an improving column to add, we switch to an iteration of an exact method. In addition, as mentioned before, to retrieve information on the intermediate dual bounds, either the pricing has to be solved to optimality or a valid Lower Bound on the optimal reduced cost must be available. Hence, we apply a hybrid strategy to solve the pricing subproblem in which, after calling qbsolv for a fixed number of CG iterations, it switches to an exact method (branch-and-bound) for one iteration. In the exact iteration, either the BQP formulation of the subproblem or the linearized version is solved by GUROBI.

As mentioned earlier, RLT is, in general, the most effective linearization approach to provide tight bounds for the QSAP variants in the literature [103, 104]. Therefore, we compare the results of our reformulation with the results of GUROBI applied to the RLT model of the AQSAP as well as the SLT model of the AQSAP. To investigate the effectiveness of the proposed reformulation, we also implement the reformulation and outer approximation suggested in Rostami et al. [49] for a class of BQP problems including the QSAP and report the results on the same instances. A brief description of the different reformulations and methods used to solve instances of the AQSAP is provided here.

BQP: The BQP model (3.10)-(3.12) solved by GUROBI.

SLT: Linearized reformulation (using SLT) of the BQP model (3.10)-(3.12) solved by GUROBI.

RLT: Linearized reformulation (using RLT) of the BQP model (3.10)-(3.12) solved by GUROBI.

OuterApproximation: Convex reformulation and outer approximation [49] applied to the BQP model (3.10)-(3.12) solved by GUROBI.

CG+BQPPricing: CG algorithm for the model (3.14)-(3.17) where the UBQP pricing is solved by GUROBI

CG+SLTPricing: CG algorithm for model (3.14)-(3.17) where the standard linearization of the pricing subproblem is solved by GUROBI.

CG+HeuristicBQPPricing: CG algorithm for model (3.14)-(3.17) where the UBQP pricing is solved by using the hybrid heuristic method described above.

CG+HeuristicSLTPricing: CG algorithm for model (3.14)-(3.17) where the UBQP pricing is solved using the hybrid heuristic in which the exact iteration solves the standard linearized pricing subproblem by GUROBI.

Note that, although we use GUROBI to solve all the obtained models in different methods, in the rest of the work, we only refer to SLT, BQP, and RLT as GUROBI methods for the sake of simplicity.

Figure 3.3 shows the results of all methods in terms of performance profile for the AQSAP. The left diagram compares all GUROBI, CG, and the OuterApproximation methods in terms of dual-bounds performance, while the right one gives the time performance comparison of the methods. According to the description provided for the dual-bound performance profile, the dual-bound ratio is between zero and one for this application. However, the diagram on the right-hand side corresponding to time performance always consists of performance ratios that are greater than or equal to 1. Clearly, for both bound and time performance profiles, a method with a larger fraction of instances with a ratio closer to 1 is preferable.

AQSAP results.

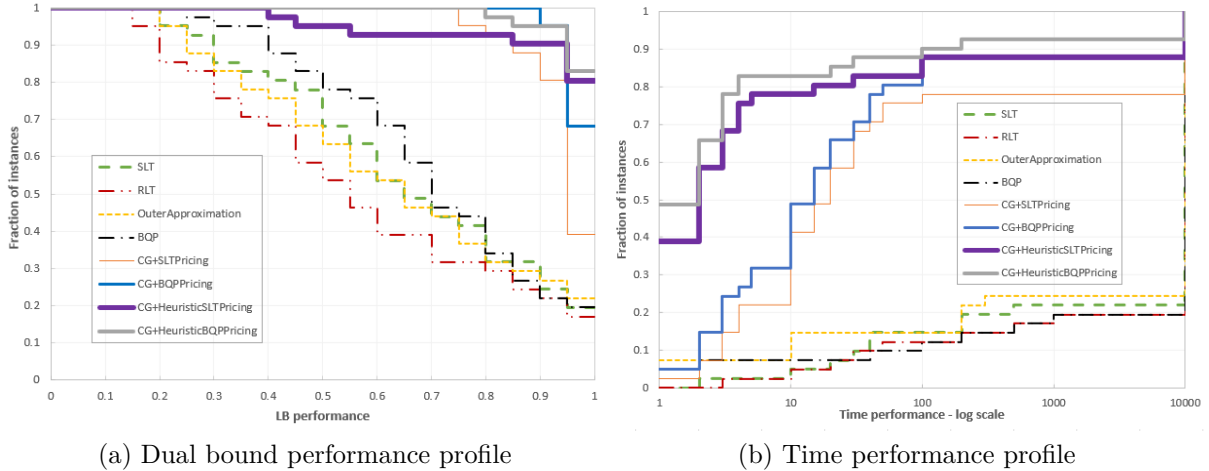


Figure 3.3 Performance profiles for AQSAP instances.

The results for the AQSAP instances are given in Figure 3.3. According to the LB performance analysis in Figure 3.3a, CG hybrid methods have the most wins in terms of providing the best LB (83% and 80% for the CG+HeuristicBQPPricing and the CG+HEURISTICBQPPricing, respectively) among all the methods, though the CG+BQPPricing outperforms the hybrid methods in a few quantiles. For instance, all the instances are solved by the CG+BQPPricing

to 90% of the best LB, while only 95% of them could reach this ratio when solved by the CG+HeuristicPricing. We also observe that in general, GUROBI solves the BQP model (BQP) slightly better than it solves the linearization models (SLT and RLT). If we seek a method that is able to achieve at least 20% of the best LB, then all of the tested methods achieve this. However, if we increase the requirement to 40%, we can observe that all the CG methods perform better than the non-CG GUROBI methods and the OuterApproximation method. Finally, looking for a method with 100% performance, the CG+HeuristicBQPPricing is the best choice.

The next analyses are related to computing time. The time performance profile in Figure 3.3b shows that the best time to find the best LB for almost 50% of instances belongs to the CG+HeuristicBQPPricing method. It also demonstrates the superiority of this method in all other quantiles. Comparing all the methods together, the GUROBI methods and the OuterApproximation method are inefficient at finding the best LBs, where the best of them in terms of time (the OuterApproximation method) provides the best time in only 7% of the cases. As another example to illustrate the superiority of CG methods, consider that using the SLT method, only 15% of the instances and using the BQP, only 12% of instances could converge to the best LB within 2 orders of magnitude of the best time.

Looking at both graphs simultaneously confirms the superiority of the CG-based methods over the GUROBI and the OuterApproximation methods, with a large gap for almost all intervals. Likewise, comparing the different methods of CG indicates that, as theory suggests, combining heuristics and exact methods to solve the pricing problems improves the results both in terms of computing time and LB for most of the instances.

3.5.2 MOT Experiments

For the MOT data association problem, we perform our tests based on a well-known benchmark, the MOT challenge datasets [105]. Our tests are performed on different instances from the same video sequence (MOT16-09) of this benchmark and produce instances by altering the input parameters in the sequence. These parameters include the number of frames in the video ($\mathcal{T}$), the maximum number of tracks (h), and the maximum number of considered adjacent frames (d). Trivially, the estimated Upper Bound h has to be increased by enlarging the number of investigated data frames to avoid missing track of any person in a real case. However, one has to consider the effects of this parameter on the growth of the problem size in the pre-estimation. In addition, we set the quadratic cost of two nodes that are more than d frames apart to zero in our implementation. We alter this parameter to demonstrate its influence on enlarging the quadratic matrix and problem size and generating various instances.

The majority of the dataset configurations in this section are based on Henschel et al. [4].

It is worth noting that to be able to compare two entire MOT algorithms, the detection of objects and the precision of estimating unary and pair-wise costs, which are normally obtained using deep learning techniques, are very crucial. However, since in the current paper, we aim to compare the proposed reformulation of data association and CG with the results of a MIP solver, computing real accurate costs is beyond the scope of our research. Therefore, although we explore the real detection of the MOT challenge dataset, we estimate naive unary and pair-wise costs for these detections based on some basic factors such as the distance between the detections. This method is exploited in several papers in data association. For example, Yarkony et al. [25] assume that the costs are provided by learning methods and are given to their algorithms.

We should remark that each frame in the MOT16-09 sequence includes 15 to 25 detections; therefore, the number of decision variables in the represented formulation of (3.20) is between $15 \times h \times \mathcal{T}$ and $25 \times h \times \mathcal{T}$. To better realize the large scale of the problem, assume we investigate a data instance related to 10 frames of a video sequence consisting of an average of 20 detections in each frame and we aim to track a maximum number of 35 people in the sequence. This instance includes 7000 decision variables in the represented BQP formulation, which generates a graph of the problem with 235 nodes in total. We show 27 generated instances from the mentioned data set in Tables A.3 and A.4 of Appendix A.

Similar to the semi-assignment problem, here we evaluate the efficiency of the star-based reformulation of MOT compared with the solver. In Section 3.4.2, we discussed that the pricing subproblem of MOT is a constrained BQP problem. So, we can apply a naturally tighter linearization, RLT, to solve the subproblems in addition to previously-used standard linearization in the QSAP. In Appendix A.1, we clarify this linearization when it is applied to the MOT formulation. Here, the solution methods are briefly introduced:

BQP: The BQP model (3.20)-(3.23) solved by GUROBI.

SLT: Linearized reformulation (using SLT) of the BQP model (3.20)-(3.23) solved by GUROBI.

RLT: Linearized reformulation (using RLT) of the BQP model (3.20)-(3.23) solved by GUROBI.

CG+BQPPricing: CG algorithm for model (3.24)-(3.27) where constrained BQP pricing is solved by GUROBI.

CG+SLTPricing: CG algorithm for the model (3.24)-(3.27) where the standard linearization of the pricing subproblem is solved by GUROBI.

CG+RLTPricing: CG algorithm for the model (3.24)-(3.27) where the linearized pricing subproblem using RLT is solved by GUROBI.

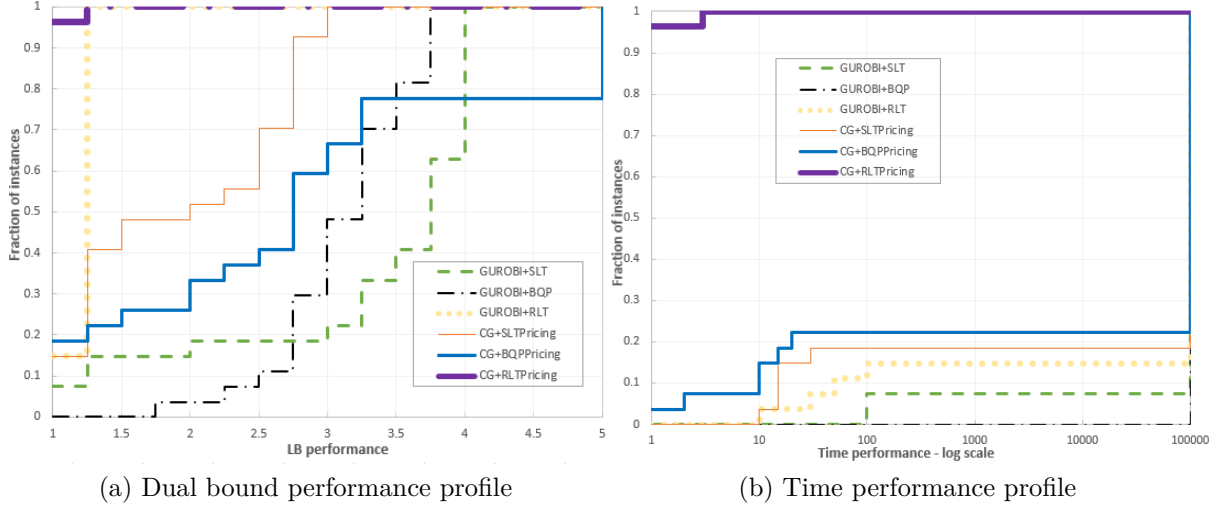


Figure 3.4 Performance profiles for the MOT problem

As discussed in the previous section, we investigate the performance of the methods through two types of performance profiles. Moreover, we report the experimental details for each instance of the problem in instance-by-instance tables in Appendix A.

According to the LB performance plot in Figure 3.4a, all the CG-based methods outperform their GUROBI-based counterparts in almost all of the intervals. The only exception occurs when it comes to comparing the CG+BQPPricing method with the BQP method. The GUROBI method demonstrates superior performance in almost 20% of the instances. As we mentioned in the definition of the dual bound performance profile, we assign a large number for the performance ratio when the method fails to provide a valid dual bound for an instance within the time limit. Given that 5 is considered a large ratio in our analyses, the figure indicates that in 22% of the instances, the CG+BQPPricing could not obtain an LB within the time limit. We identify the dual bound for these instances by NA in Table A.4. Another remark related to the LB performance graph is that all three CG methods have a number of wins that is equal to, or greater than the best GUROBI method (RLT). Evidently, when we apply the RLT to solve the pricing subproblem of the CG reformulation, the number of wins is the highest among all the methods with a large gap. Overall, the best method is the CG+RLTPricing, since it obtains the best LB for 96% of the instances and the worst LB outcome for this method for the rest of the instances is less than 1.25 times the best LB. It should be noted that the negative objective function of the data association problem in MOT is reflected in an LB performance ratio greater than 1.

Although according to Figure 3.4a the ratio of the obtained LB by the RLT to the best LB is at most 1.25, Figure 3.4b shows that its computational time is not competitive compared to CGs. More specifically, less than 20% of the test set is solved by the RLT within 2 orders of magnitude with respect to the fastest method. Hence, it still outperforms other GUROBI methods. The performance profiles in Figure 3.4 delineate the superiority of the RLT method. Evidently, when the RLT is directly applied to the BQP formulation (the BQP+RLT method), it outperforms the other GUROBIs. Moreover, when it is used as the method for solving the pricing subproblem of CG (the CG+RLTPricing method), it has better performance than the other CG methods. Considering both LB and time performance, we observe that the CG+RLTPricing obtains not only the best LB in 96% of the cases but also in the shortest time for almost all of these cases. We detail the experiments on dual bound and time, as well as UB and the parameters of the instances, later in the appendices.

Given the instance-by-instance tables in Appendix A and the performance analyses, we can infer that the star-based reformulation and the CG methodology computationally outperform the GUROBI solver in obtaining LB for the data association problem in MOT. Moreover, similar to the CG+RLTPricing, which outperforms the RLT, the other CG methods outperform their GUROBI counterparts both in terms of LB and computational time. Evidently, the CG+RLTPricing achieves the best LB in nearly all the cases and, in the situation where it stops before the time limit, it converges to an optimal solution for the vast majority of instances.

3.6 Conclusion

In this study, we investigated the generalizability and performance of the star reformulation on a large class of BQP problems, adjacent-only BQP problems. We employed the star-reformulation on two adjacent-only BQP problems with different characteristics: the AQSAP and the MOT. We compare the LB and computing time of the reformulation and CG methods with a state-of-the-art MIP solver on instances of the AQSAP and the MOT problem. According to the proposed framework, the adjacent-only class of problems inherits a special structure of the quadratic matrix which resulted in a huge improvement to solve these problems.

Considering both performance profile analyses and the instance-by-instance details of Appendix A, it can be deduced that the star-based reformulation and CG, even with a basic implementation, outperform GUROBI in terms of computing time and LB for all the instances of the AQSAP. Additionally, in a large number of the instances, given a heuristic solution for the optimal value, optimality can be proved in the root node for this problem.

CHAPTER 4 A DUAL BOUNDING FRAMEWORK THROUGH COST-SPLITTING FOR BINARY QUADRATIC COMBINATORIAL OPTIMIZATION

4.1 Introduction

Decision-making under the presence of interaction between decisions is ubiquitous, among others, in management, engineering, transportation, logistics, and computer vision. These types of problems are usually modeled as BQP which is a class of Combinatorial Optimization problems with binary variables, quadratic objective function, and linear/quadratic constraints. Solving general BQP problems is known to be NP-hard. In other words, it is unlikely to find polynomial time algorithms to exactly solve these problems.

One way to deal with the challenges of BQP is to identify patterns and structures in the objective function, constraint matrices, or instances [13, 52]. In this chapter, we employ a technique that exploits the structure of the quadratic cost matrix $\mathbf{q}$. Our methodology relies on splitting the quadratic cost into (i) in-star interactions in which the quadratic costs between adjacent edges are investigated and (ii) out-of-star interactions corresponding to quadratic costs between non-adjacent edges. This leads to a new reformulation of the BQP with exponentially many variables, each associated with a star in the graph G of Chapter 1. Inspired by the star reformulation of Chapter 3 for the adjacent-only quadratic problems, we develop Column Generation approaches to derive bounds for general BQP problems with different levels of in-star and out-of-star quadratic costs.

Based on the literature review of Chapter 2, only a few studies have tackled the difficulties of solving a specific application of the BQP models by reformulating the problem into a Mixed Integer Linear Programming formulation and hence solving it by Column Generation method [24, 25]. However, there is no general framework to propose a reformulation and CG to cover many problems in different applications of BQP.

4.1.1 Main Contributions

This study¹ provides the following contributions:

- We propose a solution framework to exploit the structure of the BQP problems to split

¹This chapter is based on the following article: Bayani, M., Rostami, B., Adulyasak, Y., Rousseau, LM. A Dual Bounding Framework for Binary Quadratic Combinatorial Optimization, Les Cahiers du GERAD ISSN. 2021 Aug;711:2440. Submitted to INFORMS Journal on Computing in July 2021 (Under 3rd revision).

the general BQP problems' quadratic costs into adjacent and non-adjacent components.

- We develop a reformulation and CG framework for general BQP problems built upon the star-reformulation introduced in Pereira et al. [20] for the Adjacent-only Quadratic Minimum Spanning Tree Problem.
- We focus on the Quadratic Semi-Assignment Problem, and conduct a comprehensive set of experiments for this problem to evaluate the performance of the method on BQP problems. By reformulating this problem, we derive a quadratic master problem and unconstrained quadratic pricing subproblems.

The rest of the chapter is organized as follows. Section 4.2 is corresponding to the proposed cost-splitting framework and then we suggest a Column Generation method to solve the reformulation. Section 4.3 is dedicated to presenting and reformulating a general BQP problem, QSAP, as an illustrative example to evaluate the cost-splitting reformulation and CG. We also display the test set and obtained results in this section. The conclusion of this study is presented in Section 4.4.

4.2 Cost-Splitting and Star-Reformulation for General BQP Problems

In Section 3.2.1 of the previous chapter, we defined some concepts on a graph G , such as adjacency set $\mathcal{A}$, star-shaped sub-graph s and set S^v as the set of all stars centered at $v \in H$. Hence, in the next section of Chapter 3, based on the star-reformulation of Pereira et al. [20] for the AQMSTP, we assessed the generalizability of this reformulation for other adjacent BQP problems.

In order to tackle the complexity of the general BQP problem introduced in Section 1.1 of the first chapter of the dissertation, by using decomposition and inspired by the idea of star-reformulation of Section 3.2 of the previous chapter, we propose a cost-splitting framework to reformulate the BQP problems. Our methodology relies on the concept of star structures to partition the objective function of (1.1) at any feasible solution $\bar{\mathbf{x}}$ into four parts: in-stars linear costs, out-of-stars linear costs, in-stars quadratic costs, and out-of-stars quadratic costs. More precisely, given a feasible solution $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{m_1}, \dots, \bar{x}_m) \in X \subseteq \{0, 1\}^m$, we can rewrite it as $\bar{\mathbf{x}} = (\bar{\mathbf{x}}^1, \bar{\mathbf{x}}^2)$ with $\bar{\mathbf{x}}^1 \in \{0, 1\}^{m_1}$ and $\bar{\mathbf{x}}^2 \in \{0, 1\}^{m-m_1}$ and where $\bar{\mathbf{x}}^1$ is related to feasible solutions of edges in set A and $\bar{\mathbf{x}}^2$ stands for feasible solutions of edges in $E \setminus A$. Therefore, the objective function of (1.1) at $\bar{\mathbf{x}}$ can be written as:

$$\sum_{e \in A} c_e \bar{x}_e + \sum_{e \in E \setminus A} c_e \bar{x}_e + \sum_{(e,f) \in A} q_{ef} \bar{x}_e \bar{x}_f + \sum_{(e,f) \in E \setminus A} q_{ef} \bar{x}_e \bar{x}_f$$

As presented in Section 3.2, the first and third terms of this equation together depict the costs of inside stars, while the last term as a quadratic cost is associated with the interaction between pairs of non-adjacent edges. Therefore, the cost-splitting reformulation of the BQP problem of (1.1) is proposed where the constraints are maintained as in (3.3)-(3.5):

$$\min \quad \sum_{s \in S} C_s \lambda_s + \sum_{e \in E \setminus A} c_e x_e + \sum_{(e,f) \in \mathcal{E} \setminus A} q_{ef} x_e x_f \quad (4.1)$$

$$\text{s.t.} \quad (\mathbf{x}, \boldsymbol{\lambda}) \in \mathcal{F}(\mathbf{x}, \boldsymbol{\lambda}) \quad (4.2)$$

$$\mathbf{x} \in \{0, 1\}^m \quad (4.3)$$

$$\boldsymbol{\lambda} \in \{0, 1\}^{|S|} \quad (4.4)$$

Although modeling BQP problems based on separating adjacent edge interactions and non-adjacent edge interactions is the principal notion behind the proposed model, each term of the objective function can potentially be eliminated based on the specific problem structure in different applications. Nonetheless, we keep the first term of the objective function as the basis of our reformulation. As an example, in the Quadratic Minimum Spanning Tree Problem [16], since the quadratic cost comprises the interactions between all pairs of edges (adjacent and non-adjacent) and the set of possible star centers consists of the complete node set V , in the objective function of the cost-splitting reformulated model of the QMSTP, we incorporate a linear term to represent the cost of stars and a quadratic term for interactions between non-adjacent edges.

In another notable example, consider the uncapacitated single allocation p -Hub median problem [106, 107] which is formulated as a BQP problem. This problem can be reformulated using our proposed star-based model, consisting of linear costs inside the possible stars and quadratic interactions between stars with different centers.

4.2.1 Column Generation Approach

As depicted in equations (4.1)-(4.4), the incorporation of stars in the reformulation of the BQP leads to an exponential number of variables denoted as $\lambda_s, s \in S$. To address this issue, we apply the Column Generation technique to effectively handle the proposed cost-splitting formulation.

Since the generic reformulation (4.1)-(4.4) is a non-convex quadratic problem, the standard CG procedure cannot be directly applied. If the model is convex, meaning that the matrix $\mathbf{q}$ is positive semidefinite, then either the underlying nature of that problem brings out a convex quadratic subproblem or convex quadratic master problem. In the case of a convex

Restricted Master Problem, the primal and dual solutions of the RMP can be obtained by state-of-the-art solvers. However, in our proposed framework, we do not restrict the definition of $\mathbf{q}$ to positive semidefinite matrices. Therefore, one has to deal with the quadratic term of the objective function. Different types of convexification [48], linearization, semidefinite programming relaxation, and BQP relaxation for block-decomposable problems [52] are alternative options for handling the BQP master problem. Nevertheless, some of these methods require additional complex constraints and variables which may adversely affect their performance. Our approach is based on the idea of converting the quadratic cost to linear cost by separating the in-star and out-of-star costs and then replacing each of them with a linear term. Here, we follow linearization techniques to transform the quadratic objective function into an equivalent MILP. We define y_{ef} as the linearized variable to replace the quadratic terms $x_e x_f$, $(e, f) \in \mathcal{E} \setminus \mathcal{A}$, impose a set of linking constraints $\mathcal{P}(\mathbf{x}, \mathbf{y})$ to guarantee $y_{ef} = x_e x_f$, and consider $\bar{S} \subseteq S$ as a subset of possible stars, so we obtain the following RMP for the LP relaxation of the problem:

$$\min \quad \sum_{s \in \bar{S}} C_s \lambda_s + \sum_{e \in E \setminus A} c_e x_e + \sum_{(e,f) \in \mathcal{E} \setminus \mathcal{A}} q_{ef} y_{ef} \quad (4.5)$$

$$\text{s.t.} \quad (\mathbf{x}, \boldsymbol{\lambda}) \in \bar{\mathcal{F}}(\mathbf{x}, \boldsymbol{\lambda}) \quad (4.6)$$

$$(\mathbf{x}, \mathbf{y}) \in \mathcal{P}(\mathbf{x}, \mathbf{y}) \quad (4.7)$$

$$\mathbf{y} \in \mathbb{R}_+^{|(e,f) \in \mathcal{E} \setminus \mathcal{A}|} \quad (4.8)$$

$$\boldsymbol{\lambda} \in [0, 1]^{|\bar{S}|}, \quad (4.9)$$

where $\bar{\mathcal{F}}(\mathbf{x}, \boldsymbol{\lambda})$ is a subset of $\mathcal{F}(\mathbf{x}, \boldsymbol{\lambda})$ restricted to $\bar{S}$.

Based on the definition of the star's cost C_s , the pricing subproblem to find the most negative reduced cost column could be a linear problem or a BQP problem.

4.3 Computational Studies

In this section, we investigate the performance of the cost-splitting idea and proposed CG on general BQP problems by considering the Quadratic Semi-Assignment Problem with different levels of quadratic cost matrix's sparsity. In the following parts, we briefly talk about the problem and its applications, hence, we represent how to implement the cost-splitting reformulation on this problem and obtain dual bounds using the CG method.

4.3.1 Quadratic Semi-Assignment Problem

In this problem, we are given a possible distinct number of servers and clients. We demonstrate the set of servers by $H = \{1, \dots, h\}$ and the set of clients by $N = \{1, \dots, n\}$. The constraints state that clients need to be assigned to servers and there is no limit for the number of assigned clients to each server. Assuming the quadratic interaction $q_{ef}, e = \{i, j\}, f = \{k, l\}$, associated with the assignment of client $i \in N$ to server $j \in H$ and client $k \in N$ to server $l \in H$ simultaneously, in addition to the linear cost $c_e, e = \{i, j\}$, corresponding to assigning each client $i \in N$ to server $j \in H$, recalling the definitions of the previous chapter, the compact model of the QSAP is:

$$\min \quad \sum_{e \in A} c_e x_e + \sum_{(e,f) \in \mathcal{E}} q_{ef} x_e x_f \quad (4.10)$$

$$\text{s.t.} \quad \sum_{e \in \delta(i)} x_e = 1 \quad \forall i \in N \quad (4.11)$$

$$x_e \in \{0, 1\} \quad \forall e \in E. \quad (4.12)$$

The QSAP has a variety of applications in the area of scheduling [108, 109] and partitioning [110]. The hub network design problem is also considered a special case of the QSAP [111]. The problem is known to be NP-hard, and solving it even for small-size instances is very time-consuming [112]. Using RLT is a common approach in the literature to solve the QSAP and there are also some studies on polynomial algorithms, heuristics, and lower bounding methods for special cases of the QSAP. We refer the reader to Saito et al. [111] and the references therein for more details.

4.3.1.1 Reformulation

To transform the QSAP according to the cost-splitting reformulation of (4.1)-(4.4), we assume that each server j represents the center of a star-shaped subgraph s . This is denoted by the binary variable λ_s , where $\lambda_s = 1$ indicates the selection of the star s . In order to identify whether server j is the center of star s or not, we introduce the parameter $B_{js} \in \{0, 1\}$. Additionally, we define the parameters $D_{is} \in \{0, 1\}$ to determine if client i is included in star s , and $D_{es} \in \{0, 1\}$ to indicate whether edge e belongs to star s or not.

Considering the out-of-star interactions in the QSAP, the reformulation involves quadratic terms. Therefore, utilizing the notation provided above and referring to the formulation

(4.5)-(4.9), the linearized RMP is expressed as follows:

$$\text{[RMP-QSAP]:} \quad \min \quad \sum_{s \in \bar{S}} C_s \lambda_s + \sum_{(e,f) \in \mathcal{E} \setminus \mathcal{A}} q_{ef} y_{ef} \quad (4.13)$$

$$\text{s.t.} \quad \sum_{s \in \bar{S}} B_{js} \lambda_s \leq 1 \quad \forall j \in H \quad (4.14)$$

$$\sum_{s \in \bar{S}} D_{is} \lambda_s = 1 \quad \forall i \in N \quad (4.15)$$

$$\sum_{s \in \bar{S}} D_{es} \lambda_s = x_e \quad \forall e \in A \quad (4.16)$$

$$(\mathbf{x}, \mathbf{y}) \in \mathcal{P}(\mathbf{x}, \mathbf{y}) \quad (4.17)$$

$$0 \leq x_e \leq 1 \quad \forall e \in A \quad (4.18)$$

$$\mathbf{y} \in \mathbb{R}_+^{|(e,f) \in \mathcal{E} \setminus \mathcal{A}|} \quad (4.19)$$

$$\boldsymbol{\lambda} \in [0, 1]^{|\bar{S}|}, \quad (4.20)$$

where y_{ef} are the linearization variables.

Constraints (4.14) ensure that there is a maximum limit of one selected star for each center j . Constraints (4.15) are set partitioning linking constraints that impose the connection between the stars and the clients. On the other hand, constraints (4.16) are linking constraints enforcing that if an edge is chosen in an optimal solution, it can only be included in one selected star.

4.3.1.2 Column Generation

Starting from a subset of stars ($\{s | \lambda_s = 1\}$) which are feasible with respect to the constraints of the reformulation as initial columns, the algorithm solves the [RMP-QSAP] restricted to the current set of stars in each iteration. The corresponding dual solutions construct one pricing subproblem for each server $j \in H$, which aims to find a star related to j with the minimum reduced costs. In the next iteration of the algorithm, these stars are added to [RMP-QSAP] as the new columns to possibly improve the objective value of the master problem. The algorithm terminates when there are no more columns with negative reduced costs to be added.

Let π_j , ρ_i and γ_{ij} be the dual solutions corresponding to constraints (4.14) to (4.16), respectively. For each server j we consider a set of incident edges to that server. Therefore, we can use them to rewrite the dual solutions $\gamma_{i,j}$ as γ_e , where $e = \{i, j\}$. In addition, ρ_i is not dependent on j and is fixed for all star centers, so we use $\rho_e, e = \{i, j\}$ instead of ρ_i in our formulation to keep it consistent with the rest of the terms. The star with minimum reduced

cost can be found by solving the following pricing subproblems on graphs, one for each server j :

$$\min \sum_{e \in \delta(j)} (c_e - \rho_e - \gamma_e) z_e + \sum_{e, f \in \delta(j)} q_{ef} z_e z_f - \pi_j \quad (4.21)$$

$$\text{s.t. } z_e \in \{0, 1\} \quad \forall e \in \delta(j), \quad (4.22)$$

where binary decision variable $z_e, e = \{i, j\}$, indicates if client i is part of the star centered at server j .

4.3.2 Experiments

This section presents a comprehensive experimental analysis conducted to evaluate the effectiveness of our proposed framework. The evaluation is performed on test instances of the QSAP by comparing the dual bounds and computing time obtained by the cost-splitting framework and a commercial MIP solver, GUROBI. We aim to assess to what extent employing the cost-splitting framework is worthwhile for a BQP problem that has out-of-star quadratic interactions in addition to in-star interactions.

We compare the cost-splitting and CG framework with two commonly used methods in the literature of BQP: Standard Linearization Technique and Reformulation Linearization Technique. Examples of linearizations are illustrated in Appendix A. Additionally, we compare our results with one of the most effective exact methods proposed for QSAP [49].

For benchmarking, we utilize GUROBI version 9.0.1. Additionally, we directly solve the BQP models using GUROBI. It is important to note that through the CG procedure, we consider various reformulations that result in different formulations of pricing subproblems and report all the related results. The considered time limit for solving each instance is 3 hours. Note that, we choose the root node dual bound of the RMP as a reliable indicator to compare with the dual bound obtained from the solver. CG's computation time at the root node is also compared with GUROBI's.

Similar to the experiments in Chapter 3, we assess the trade-off between computation time and the quality of the dual bound using performance profiles [94, 50]. Dual bound performance profile and time performance profile are elaborated in detail in Section 3.5. Investigating these two profiles provide a comprehensive understanding of the performance of each solution strategy in comparison to the others.

We should note that all the assumptions and remarks that have been considered in the experiments of Chapter 3 are also valid for the experiments of this chapter to have a fair

understanding of the effects of adding non-adjacent quadratic costs to the Quadratic Semi-Assignment Problem.

The implementation of all algorithms and models was carried out using the Python programming language. The experiments were conducted on a shared cluster equipped with a 3.05GHz four-core processor and 128GB RAM. The cluster was running on Linux 7.8.

In the following section, we provide information regarding the test instances, parameter configurations, and computational results for the QSAP. For more detailed results refer to the instance-by-instance tables available in Appendix B.

There is a limited available instance in the literature of QSAP, so most of the papers are conducted on randomly generated data instances [101, 49]. Moreover, our main objective was to examine instances with different levels of quadratic cost matrix sparsity. Hence, we generated random instances, where the procedure is the same as Section 3.5.1 and the maximum size of the server's set is 14 and the client's set is 50.

To investigate the impacts of the sparsity and structure of the quadratic matrix on the performance of cost-splitting framework, we run the experiments on 4 randomly-generated data sets with different sparsity of the quadratic matrices, each comprised of 41 instances. We embark on our experiments by adding out-of-star interactions incrementally to observe the effects. The quadratic matrix of the first problem consists of 10% out-of-star costs in addition to the in-star interactions, while the next problems include 15%, 20%, and 25% out-of-star interactions, respectively. We should take into account that, in real-life applications of BQP, the quadratic matrix is mostly very sparse. Therefore, adding just 10% out-of-star interactions on top of the in-star interactions results in a fairly dense quadratic matrix.

As indicated in Section 4.3.1, the pricing subproblem is an unconstrained BQP. In some of the experiments, we address the subproblem using an open-source solver called qbsolv [102]. In the case that the heuristic solver fails to discover an improving column, we switch to an iteration of an exact method. Additionally, to obtain information regarding the intermediate dual bounds, it is necessary to solve the pricing either to optimality or obtain a valid Lower Bound for the optimal reduced cost. Therefore, we employ a hybrid strategy to solve the pricing subproblem. For a single iteration, qbsolv switches to an exact method (branch-and-bound) after a predetermined number of CG iterations. In this exact iteration, either the BQP formulation of the subproblem or its linearized version is solved using GUROBI.

We chose the standard linearization technique to construct the set $\mathcal{P}(x, y)$ in constraints (4.17).

Similar to the experiments of the AQSAP in Chapter 3, for the QSAP, we compare the results

of the reformulation and solving methodologies with different linearizations, in addition to the recent method proposed for QSAP in Rostami et al. [49]. In the following, we present a concise overview of the methods compared for tackling instances of the QSAP.

BQP: The BQP model (4.10)-(4.12) solved by GUROBI.

SLT: Linearized reformulation (using SLT) of the BQP model (4.10)-(4.12) solved by GUROBI.

RLT: Linearized reformulation (using RLT) of the BQP model (4.10)-(4.12) solved by GUROBI.

OuterApproximation: Convex reformulation and outer approximation [49] applied to the BQP model (4.10)-(4.12) solved by GUROBI.

CG+BQPPricing: CG algorithm for the model (4.13)-(4.20) where the UBQP pricing is solved by GUROBI

CG+SLTPricing: CG algorithm for model (4.13)-(4.20) where the standard linearization of the pricing subproblem is solved by GUROBI.

CG+HeuristicBQPPricing: CG algorithm for model (4.13)-(4.20) where the UBQP pricing is solved by using the hybrid heuristic method described above.

CG+HeuristicSLTPricing: CG algorithm for model (4.13)-(4.20) where the UBQP pricing is solved using the hybrid heuristic in which the exact iteration solves the standard linearized pricing subproblem by GUROBI.

We should remark that in the rest of the chapter, we refer to SLT, BQP, and RLT as GUROBI methods.

4.3.3 Results

The results for the QSAP instances are shown in Figures 4.1 to 4.4 where to increase the density of the quadratic matrix, we add more out-of-star interactions to the problem each time to generate our data sets. All the plots on the left are dedicated to dual-bounds performance profiles, while all the plots on the right are corresponding to the time performance profile. In terms of both bounding and time performance profiles, it is preferable to have a method with a higher proportion of instances with a ratio approaching 1. For more details on how performance profiles work, we refer the reader to Section 3.5 of the previous chapter.

We start our investigations by adding only 10% out-of-star interactions (quadratic costs) in addition to the in-star interactions. Considering the first plot, 4.1a, which is related to the LB,

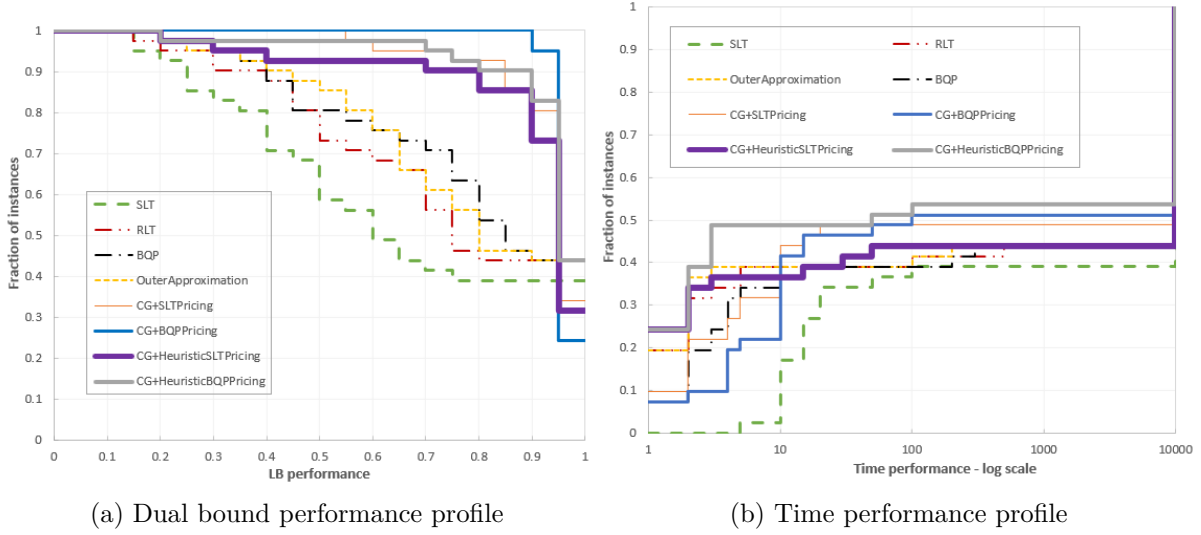


Figure 4.1 QSAP-10% out-of-star density

all the CG methods are above the other methods in the majority of the quantiles representing the superiority of the CG. In a closer look, we find that the CG+HeuristicBQPPricing has the greatest number of instances with the best LB among all the methods (44%); however, the CG+BQPPricing outperforms the hybrid counterpart in many intervals. This figure also demonstrates that the BQP and the OuterApproximation methods are superior to the other GUROBI methods while they still have a huge LB performance gap compared to all the CG methods.

Nevertheless, in the next LB performance profile represented in Figure 4.2a, all GUROBI methods and the OuterApproximation outperform CGs in the interval $[0.95, 1]$. Comparing the CG methods with one another in this figure displays that the CG methods with exact pricing generally outperform heuristic pricing methods. Considering this figure and the next LB performance figures, 4.3a and 4.4a, the number of wins for the GUROBI and the OuterApproximation methods is more than for CG methods. Indeed, adding more out-of-star interactions incrementally to the data sets in Figures 4.3a and 4.4a results in enhancing the performance of the GUROBIs and the OuterApproximation method. However, even for the densest quadratic matrix in Figure 4.4a, the CG methods still outperform non-CG methods in some quantiles. For example, we can observe the superiority of both CG+BQPPricing and CG+SLTPricing to GUROBI methods when a ratio smaller than 70% is considered and to the OuterApproximation method when the ratio is smaller than 60%. An overview of LB performance profiles for all of the data sets in the QSAP demonstrates that CG methods are more robust because they can achieve satisfactory LB performance for most of the instances.

While if we need higher LB ratios, we aim to use GUROBI and the OuterApproximation. For instance, in the case of 20% out-of-star density and for a minimum requirement of 60% LB performance, we would opt for the CG+BQPPricing as all the instances reach this ratio of the best bound when they are solved using this method. In contrast, the best non-CG method, the OuterApproximation, satisfies this requirement in only 90% of the problem instances. Assessing non-CG methods, the plots suggest increasing the performance of both RLT and the OuterApproximation compared to BQP and SLT by adding more out-of-star costs, to an extent that they outperform BQP and SLT in most of the quantiles in Figure 4.4a.

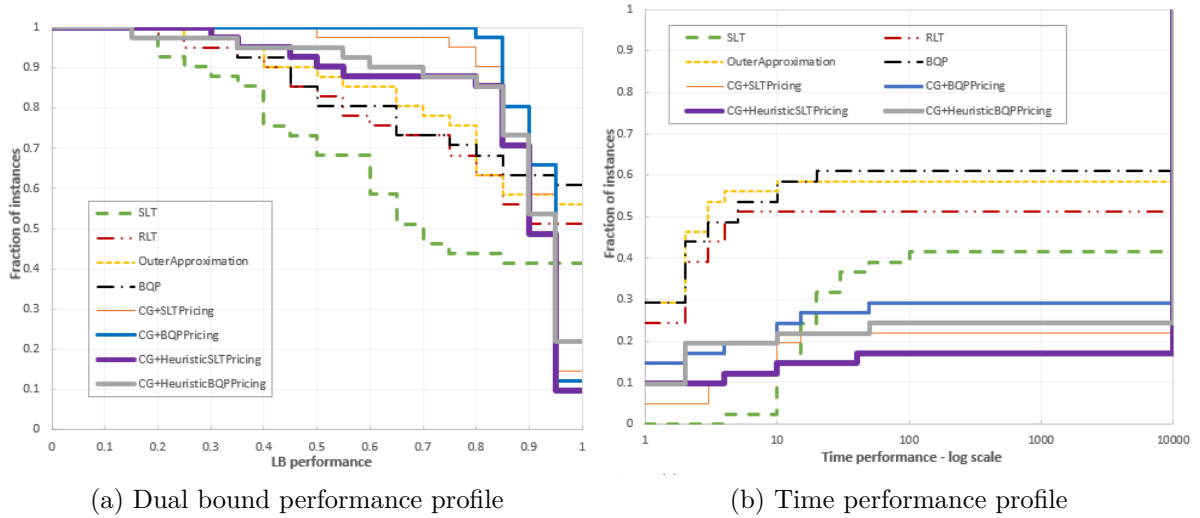


Figure 4.2 QSAP-15% out-of-star density

The second analysis is related to the time performance profiles. Looking at the plot associated with time performance in Figure 4.1, when we add out-of-star interactions, the GUROBI methods and the OuterApproximation move upside of the figure and get closer to the CG methods. This is due to the fact that the speed of the GUROBI and the OuterApproximation methods increases when the instances consist of some out-of-star interactions in addition to the in-star interactions. However, still, in Figure 4.1b, most wins belong to CG methods. Starting from Figure 4.2b, SLT, RLT, and BQP methods outperform all CG methods in all quantiles. Comparing non-CG methods together, similar to the LB analysis, the OuterApproximation outperforms the GUROBI methods in terms of time and it is superior even when we add more out-of-star costs.

It is interesting to note that, based on the last vertical lines of Figure 4.1b, around 47% of the instances could not obtain the best LB when they are solved by the best CG method (CG+HeuristicBQPPricing), while this number is 57% for the best GUROBI method and the

OuterApproximation method. Although inefficiency at finding the best LB increases for CG methods by adding more out-of-star costs, GUROBI methods can reach the best LB for more instances, in more dense matrices, due to their speed improvement. Regarding the vertical lines for the BQP method in Figures 4.2b and 4.3b, our interpretation is that this method could not obtain the best LB in the time limit for more than 35% of instances, whereas it obtained the best LB in the best time for the other 65% of instances. Moreover, we observe that the advantages of using the hybrid methods are weakened for more dense quadratic matrices. As an example, Figure 4.4b shows that the CG+HeuristicBQPPricing method outperforms the CG+BQPPricing in only 4% of the cases in terms of time performance.

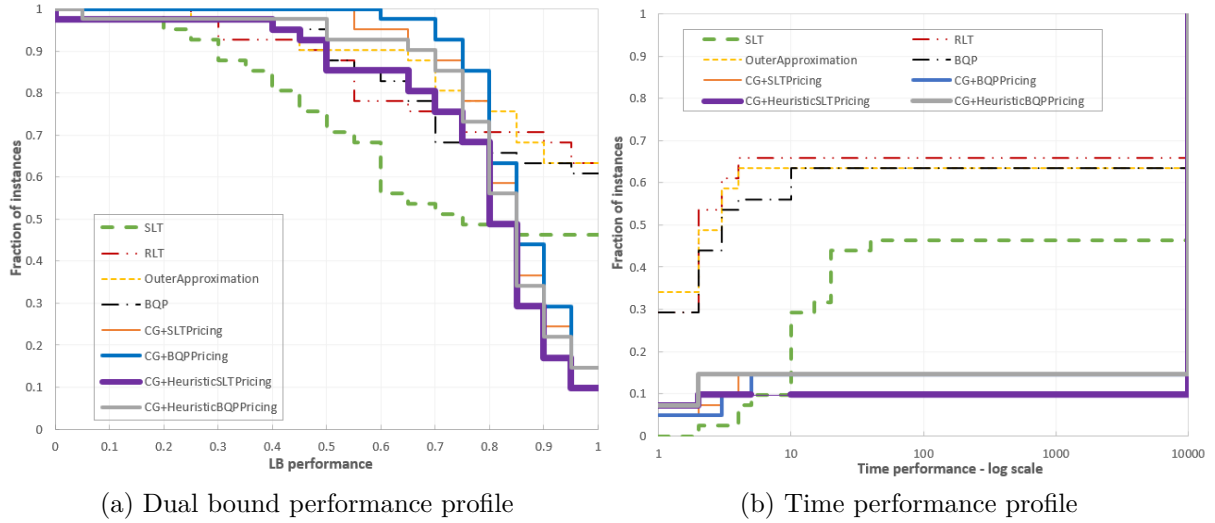


Figure 4.3 QSAP-20% out-of-star density

Looking at the trend in all of the time performance profiles in Figures 4.1b to 4.4b, we observe that the GUROBI methods and the OuterApproximation method generally outperform CG methods when the quadratic cost matrix of QSAP is denser. For instance, in Figure 4.4b where we have 25% out-of-star quadratic costs, in approximately 63% of the instances, the best GUROBI method (RLT) is within one order of magnitude with respect to the best time. Nevertheless, when we solve the instances using the best CG method (CG+BQPPricing), only around 18% of the cases are within this order. We should remark that, in our time performance analysis, we do not reflect the computing time of the instances which could not obtain the best LB. In an extreme example, suppose that CG stopped in a few seconds with a high ratio of the best LB while GUROBI found the best LB in 3 hours. In this situation, since CG did not yield the best LB, we assign a very large number (10000) to its time performance. If the GUROBI method is the only method that obtains the best LB, we consider its time performance equal to 1.

Considering both performance profile analyses and the instance-by-instance details of Appendix B, it is observable that the performance of the cost-splitting framework is reduced by augmenting the out-of-star quadratic matrix and it might not be highly effective when the matrix is fairly dense. It should be noted that since the majority of real applications of the BQP consist of sparse quadratic matrices [113], 10% to 25% out-of-star interactions in addition to in-star interactions provide the proper condition to investigate the effects of having dense matrices in our proposed reformulation.

One notable remark is that, although the performance of CG methods appears to deteriorate when solving the instances with more out-of-star costs, their performance to solve larger instances is still generally good. The results are shown in instance-by-instance tables in Appendix B suggest that for the majority of large instances, the CG methods generally provide the best bounds when the time limit is reached. In this case, GUROBI and OuterApproximation methods' dual bounds are relatively weak. In practice, we may choose CG methods over non-CG methods for even denser quadratic matrices. For example, when we have 15% out-of-star density, all the instances solved by the CG+BQPPricing provide an overall LB with the ratio of at least 80% of the best LB, while this ratio is 30% when using the best GUROBI method, BQP. Although GUROBI and OuterApproximation methods can solve small instances quite efficiently and obtain the best bounds, they still have a huge optimality gap when solving large instances after 3 hours.

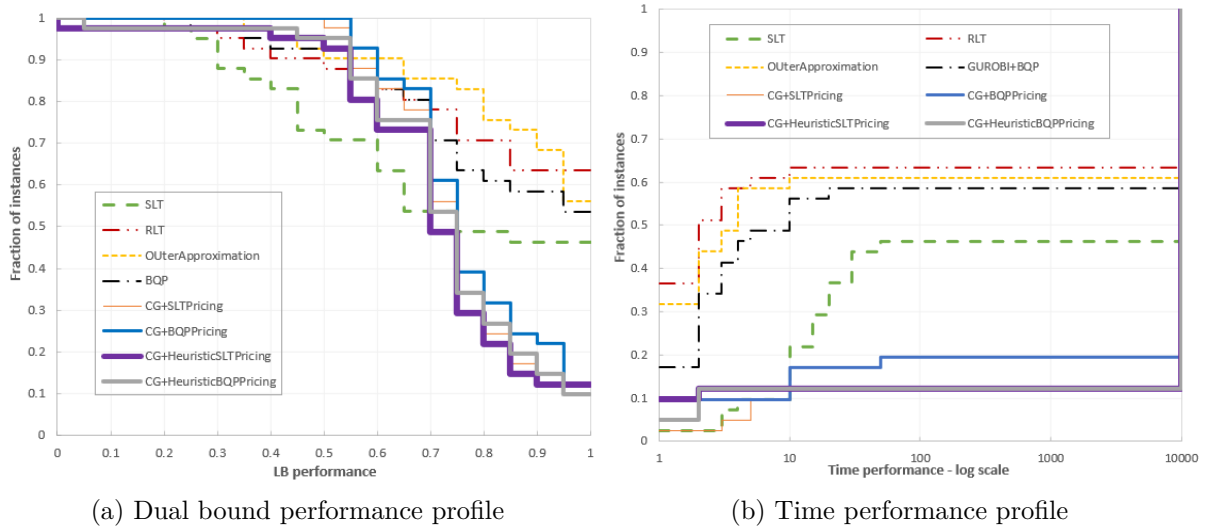


Figure 4.4 QSAP-25% out-of-star density

4.4 Conclusion

In this study, we developed a cost-splitting reformulation framework and solution methodology using CG for general BQP problems. This framework, based on in-star and out-of-star interactions between pairs of edges in the BQP problem’s graph, exploits the quadratic matrix structure. To evaluate the efficiency of our framework, we perform experiments on the QSAP. We compare the Lower Bound and computing time of the reformulation and CG methods with a state-of-the-art MIP solver on instances of this problem.

One notable outcome of this study is that when out-of-stars quadratic costs are added to the problem incrementally, the potential of the framework to compete with the solver decreases. Nevertheless, even in the case of QSAP with a fairly dense out-of-star quadratic matrix, the cost-splitting and CG methods still obtain promising results for some instances. Particularly in larger instances, where all the tested methods meet the time limit, CG methods outperform the MIP solver in many cases.

CHAPTER 5 A DATA-DRIVEN METHOD FOR CONSTRAINT CUSTOMIZATION IN OPTIMIZATION MODELS

5.1 Introduction

Optimization models are employed in various industrial applications to help planners make better decisions. However, most common models, when embedded in software-based decision support systems, fail to capture all the subtleties or practical requirements of the specific operational settings they are deployed into. Hence, planners often need to manually modify the output solutions (even when they are optimal from the model's point of view) to meet informal operational rules and preferences [26]. These post-optimization adjustments are mostly human-driven, susceptible to error, and time-consuming. Furthermore, as the business rules have the potential to change over time and be influenced by the preferences of different decision-makers within the company, the process of customization becomes time-consuming and may need to be repeated. The main drawback of this type of customization is that the software transforms into a customized decision support system for each individual customer, which is not an ideal solution approach in today's information technology industry.

As we briefly mentioned in Section 2.3.2.2, Hewitt and Frejinger [26] construct a prediction model to capture the underlying rules that decision-makers take into account when intervening in the optimization process and altering the optimal solutions. They suggest that this customization can be integrated into the optimization model in the first place. By leveraging simple machine learning methods, in addition to the traditional optimization models, they guarantee to satisfy many of the operational rules which are specific to each company. Building upon their study, in this chapter, our main idea is to develop a more sophisticated prediction model rather than using simple regression which has been used in Hewitt and Frejinger [26]. Note that the corresponding constraints from this model can be still easily added as extra constraints to a general mixed integer linear program (MILP) without any need to deal with non-linearity. We should remark that we focus on customizing MILP class of problems since most of the fundamental optimization models used in industries stem from this class.

5.1.1 Main Contributions

In the current study¹, the contributions are as follows:

- We showcase how regularized linear regression and decision trees can enhance the efficacy of the data-driven constraint customization framework.
- We present a MILP model that integrates customization constraints that take the form of decision trees, enabling us to learn and incorporate non-linear or logical customization constraints.
- In addition to linear programming problems, we empirically validate the performance of the framework on the binary knapsack problem and Nurse Rostering Problem where various combinations of concealed operational rules are simulated.

The rest of the chapter is organized as follows: The next section provides an overview of the related literature. In Section 5.2, we present the proposed framework of [26] and give more precisions on our contributions in that context. Section 5.3 is dedicated to presenting illustrative examples where we implement the approach to two industrial problems and report the obtained results. Finally, in Section 5.4 we discuss this study's conclusions.

5.2 Description of the Solution Framework

In this section, we first briefly explain the framework introduced in Hewitt and Frejinger [26] which is built based on the following three main steps:

1. Defining the general structure of the problem, referred to as the General Optimization Problem.
2. Training a machine learning model based on historical data of the previous optimal solutions and modifications applied to each decision variable of the GOP.
3. Embedding the trained models in the GOP as soft constraints and providing an augmented model, which we refer to as the Adopted Optimization Problem.

¹This chapter is based on the following article: Bayani, M., Adulyasak, Y., Rousseau, LM. A data-driven method for constraint customization in optimization models (will be submitted to INFORMS Journal on Optimization)

We assume the decision support software company knows the GOP of the customer, which is a traditional MILP for each decision point t :

$$\max \quad \mathbf{r}^t \mathbf{x} \quad (5.1)$$

$$\text{s.t.} \quad \mathbf{A}^t \mathbf{x} \leq \mathbf{b}^t \quad (5.2)$$

$$\mathbf{x} \in \mathcal{X}, \quad (5.3)$$

where $\mathbf{r}^t$, $\mathbf{A}^t$ and $\mathbf{b}^t$ are the parameters of the problem. We also assume historical data for a set of time periods ($t : 1 \dots T$) is available. We then compare the optimal solution ($\bar{\mathbf{x}}^t$) and the actionable plan ($\bar{\mathbf{y}}^t$) at decision points t . The assumption is that the customer has applied business rules to the optimal solutions that altered them to actionable plans. These business rules act as side constraints $\mathbf{G}^t \mathbf{x} + \mathbf{H}^t \mathbf{v} \leq \mathbf{u}^t$ that are missing in the GOP. These constraints consist of the GOP regular variables ($\mathbf{x}$) and possible extra decision variables ($\mathbf{v} \in \mathcal{V}$) such as ones related to if-then rules or precedence in scheduling. It should be remarked that the matrices and vectors are shown in bold text throughout this work.

The second task revolves around training a model based on historical data. The objective is to keep the AOP as a MILP to maintain the existing solution pipeline. We thereby opt for machine learning prediction models that are linearizable and tractable. Hewitt and Frejinger [26] employ linear regression in their proposed framework. For each element of the variables of the optimization problem, the regression equation is learned from the data set $(\bar{\mathbf{x}}, \bar{\mathbf{y}}_i)$ as follows:

$$y_i - \left(\hat{\beta}^i + \sum_{j \in N} (\beta_j^i x_j) \right) = 0 \quad \forall i \in N : 1, \dots, n, \quad (5.4)$$

Where $\hat{\beta}^i$ and β^i are the interception and coefficients of the i th equation, respectively, and the least squares method is used throughout their work.

The third step of the framework proposed in Hewitt and Frejinger [26] is to embed the learning model in the GOP to create the AOP. It is possible to embed a machine learning model into an optimization problem only if a suitable encoding has been defined. They define δ_i as the difference between each pair of predictions of the executable plan (y_i) and optimal

solution (x_i) . The AOP is therefore represented as following for each decision point t :

$$\max \quad r^t \mathbf{x} - c\Delta \quad (5.5)$$

$$\text{s.t.} \quad A^t \mathbf{x} \leq b^t \quad (5.6)$$

$$y_i - (\hat{\beta}^i + \sum_{j \in N} (\beta_j^i x_j)) = 0 \quad \forall i \in N : 1, \dots, n \quad (5.7)$$

$$\delta^i \geq y_i - x_i \quad \forall i \in N : 1, \dots, n \quad (5.8)$$

$$\delta^i \geq x_i - y_i \quad \forall i \in N : 1, \dots, n \quad (5.9)$$

$$\Delta = \sum_{i \in N} \delta^i \quad \forall i \in N : 1, \dots, n \quad (5.10)$$

$$\mathbf{x} \in \mathcal{X} \subseteq \mathcal{R}^n, \mathbf{y} \subseteq \mathcal{R}^n \quad (5.11)$$

$$\delta^i \geq 0 \quad \forall i \in N : 1, \dots, n \quad (5.12)$$

$$\Delta \geq 0, \quad (5.13)$$

where $\hat{\beta}^i$ and β^i are fixed learned parameters, and Δ is a variable to demonstrate the sum of all differences between predictions and solutions. Moreover, the objective function is designed to stimulate the AOP to obtain solutions close to predictions of executable plans. c is a given penalty for the AOP. This penalty term balances the importance of the original GOP objective and the difference between predictions and solutions and needs to be tuned for each problem. By solving the AOP, we obtain an optimal solution $(\mathbf{x})$. The promise is that the new solution satisfies the business rules for more instances than the original solutions.

This framework is devised assuming that the underlying problem can be modeled as an optimization problem and that the parameters are known in advance. Additionally, a dataset of historical records, comprising optimal solutions (intended decisions) and executed plans (actual decisions) exists. To evaluate the approach's effectiveness, the following additional steps need to be considered in this study: generating data and validating the framework.

As shown in the framework depicted in Figure 5.1, the first step is data generation. The experiment is a controlled setting since, in this study, the true data generation is known. The collected data is partitioned into two subsets: the training set and the test set. The former is utilized to acquire knowledge on the linear representation of a mapping. With the latter, we evaluate the effectiveness of these learned mathematical entities.

For each instance of the problem, the parameters are generated randomly and the constructed problem is solved to optimality. Once the optimal solutions are obtained, the predefined rules and preferences are applied, creating the new altered solutions. In the next step, a machine learning model is trained and embedded into the optimization model. As such, executable

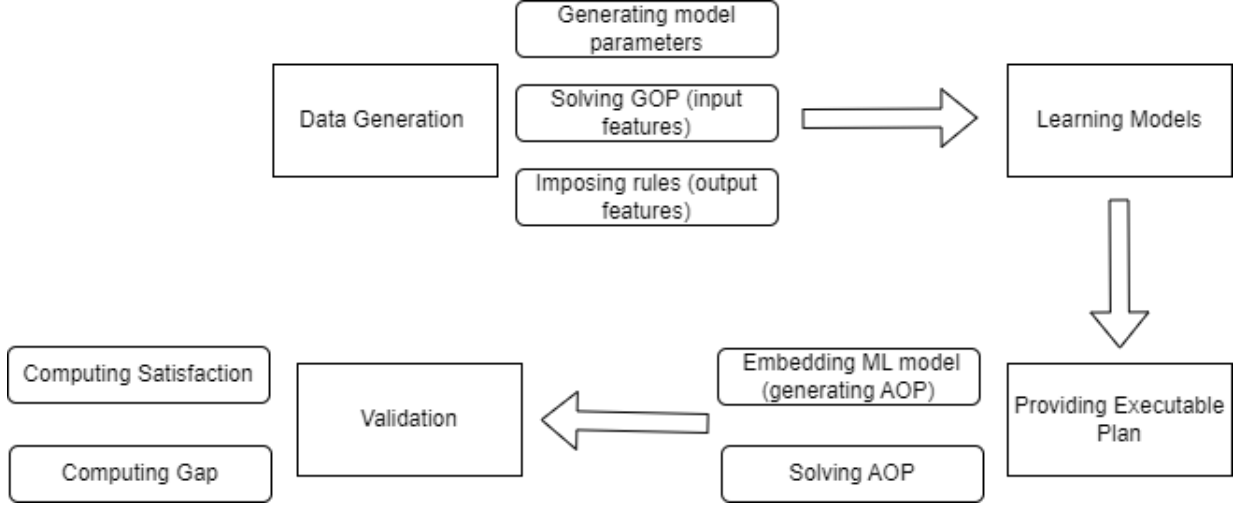


Figure 5.1 Framework Overview

plans are generated by solving the new optimization problem (AOP). In order to validate the method, we evaluate two criteria. Firstly, we measure the percentage of feasible instances regarding the unmodeled rules after solving the AOP. Secondly, assuming the rules are known, we compute the difference between the cost of adding the learned constraints to the model and the cost of adding the rules directly to the model. This performance measure is associated with the degradation of the objective function.

In the following, we explain the contributions of this study in the learning phase of the framework. In Section 5.2.1, we briefly describe what kind of regularizations have been used in linear regression. Section 5.2.2 represents how we model and incorporate decision trees instead of linear regression in the framework.

5.2.1 Regularized Linear Regression

Hewitt and Frejinger [26] use a sparse matrix of coefficients in generating the AOP by removing the coefficients with p -value of higher than 0.05. We also believe that using a sparser model has some advantages. The promise is that using regularization leads to obtaining a trained model only based on more important features instead of all of them. So, each variable y is influenced by a limited number of variables x . We assume that by regularization, we reduce the number of variables in each constraint, which is more likely to happen in reality. This model results in a more easily explainable and computationally tractable AOP [114]. We explore a regularized version of the described linear regression to pursue this idea. As such, we apply lasso regression, ridge regression, and elastic Net (eNet). While the regularized

linear regression equation is the same as the linear regression, lasso regression extends it by adding an $L1$ -norm penalty to the model's coefficients ($\lambda \|\boldsymbol{\beta}\|_1$). The shrinkage parameter, λ , controls the sparsity of the coefficients matrix by setting less important features to zero. On the other hand, the $L2$ -norm penalty term in ridge regression limits the magnitude of weights and shrinks them toward zero but does not set them exactly to zero. Elastic Net employs a combination of $L1$ and $L2$ regularization, which can help overcome some of the limitations of each separate regularization.

5.2.2 Decision Trees

In this study, we suggest using decision trees as a powerful machine learning technique to predict latent (possibly nonlinear) business rules, as they are also compatible with MILP. In decision trees, the feature space is partitioned into subspaces, and each subspace has its specific fitted prediction model. Therefore, a piece-wise linear model can be constructed to approximate nonlinear rules. Regression trees can be used when dealing with fractional variables whereas classification trees would be beneficial with the presence of discrete variables [115]. Inspired by the studies of Bertsimas and Dunn [116] and Dunn [115], one can embed decision trees in an optimization model by defining auxiliary binary variables $\mathbf{z}$, and big- M

constraints:

$$\max \quad \mathbf{r}^t \mathbf{x} - c\Delta \quad (5.14)$$

$$\text{s.t.} \quad \mathbf{A}^t \mathbf{x} \leq \mathbf{b}^t \quad (5.15)$$

$$y_i - (\hat{\beta}^{il} + \sum_{j \in N} (\beta_j^{il} x_j)) \leq M(1 - z^{il}) \quad \forall i \in N, \quad \forall l \in L \quad (5.16)$$

$$y_i - (\hat{\beta}^{il} + \sum_{j \in N} (\beta_j^{il} x_j)) \geq -M(1 - z^{il}) \quad \forall i \in N, \quad \forall l \in L \quad (5.17)$$

$$\sum_{j \in N} (h_j^{im} x_j) \leq g^{im} + (1 - z^{il}) \quad \forall i \in N, \quad \forall l \in L, \quad \forall m \in \text{Left}(l) \quad (5.18)$$

$$\sum_{j \in N} (h_j^{im} x_j - h_j^{im} \epsilon_j) \geq g^{im} - (1 - \epsilon_{max})(1 - z^{il}) \quad \forall i \in N, \quad (5.19)$$

$$\forall l \in L, \quad \forall m \in \text{Right}(l)$$

$$\sum_{l \in L} z^{il} = 1 \quad \forall i \in N \quad (5.20)$$

$$\delta^i \geq y_i - x_i \quad \forall i \in N \quad (5.21)$$

$$\delta^i \geq x_i - y_i \quad \forall i \in N \quad (5.22)$$

$$\Delta = \sum_{i \in N} \delta^i \quad \forall i \in N \quad (5.23)$$

$$\mathbf{x} \in \mathcal{X} \subseteq \mathcal{R}^n, \mathbf{y} \subseteq \mathcal{R}^n, \mathbf{z} \in \{0, 1\}, \quad \delta^i \geq 0 \quad \forall i \in N, \quad \Delta \geq 0 \quad (5.24)$$

We train n separate decision trees ($i : 0, \dots, n-1$), since the problem is a multi-output classification. It is important to note that, in practice, training the model for all of the n variables is not necessarily required. Instead, one can train decision trees (or any other machine learning models) for only the variables that can vary between the executed plans and the optimal solutions. Assuming each tree consists of $|L|$ leaf nodes, we define z^{il} to represent the selection of leaf node $l \in L$ for the output i . At each leaf node l in each tree, we make the prediction using equations (5.16) and (5.17), where $\hat{\beta}^{il}$ and β^{il} are learned parameters and M is a sufficiently large constant. Here, (5.16) and (5.17) are represented in the most general format when we have a linear regression prediction for each leaf node. Nonetheless, they can be merely constant or 0-1 predictions for integer programming through a classification tree. Constraints (5.18) and (5.19) enforce branching by traversing each leaf node's left ancestors and right ancestors, respectively. m indicates intermediate nodes. If the learning method applies hyperplane splittings, several coefficients h can have nonzero values simultaneously. In this study, we only investigate the default machine learning method of Scikit-learn 0.21.2, CART [117]. In this library, univariate splitting is used for regression and classification trees. Hence, each node has only one x_j with a non-zero coefficient. The

constant values g are thresholds of branching in the trees and ϵ and ϵ_{max} are defined to generate compatible inequality constraints for MILP solvers. Moreover, selecting only one leaf node for each output of every problem instance is guaranteed according to constraints (5.20).

Note that, although applying a random forest [118] instead of a single tree may result in a more accurate predictor, embedding a random forest in an optimization problem requires adding many extra variables and constraints. This greatly expands the size of the problem.

5.3 Test Problems and Computational Experiments

This section presents an experimental study to investigate the performance of the framework. To this end, we consider the fractional and binary knapsack problems and the Nurse Rostering Problem. We evaluate the framework on the test instances of these problems in terms of both feasibility of the solutions, and the degradation in the objective function. In each part, we briefly define the test problem, and the data generation procedure is later explained. The computational results are detailed in the last section for each test problem.

GUROBI [119] version 9.0.1 is used as our benchmark MIP solver for solving the GOP and the AOP models. All of the algorithms and models were implemented in Python programming language. The machine learning models are trained using Scikit-learn [120] 0.21. We divide the data into a training set (80%) and a test set (20%). Tenfold cross-validation and GridSearch are performed for hyper-parameter tuning for all models. In lasso and ridge regressions, we tune the shrinkage parameter, while in elastic Net, another parameter, $l1_ratio$, which combines lasso and ridge penalties and assigns weights to them, is also tuned. The GridSearch hyper-parameter tuning process for shrinkage parameter is performed over a discrete set in the range of 0.0001 and 10. In addition, the $l1_ratio$ tuning is performed using a step-size approach over the range of $0 \leq l1_ratio \leq 1$. The Scikit-learn library is utilized for decision trees as well. For decision trees, we optimize the maximum depth and the criterion as measures of the splits' quality. In our hyper-parameters search procedure, we tune the model where the maximum depth is changing over the range of 1 to 10, and the criterion is selected between gini and entropy.

5.3.1 Linear and Binary Knapsack problems

We consider the problem of loading different products in a container with capacity b . Each product i has a size of a_i and yields a profit of r_i and the objective is to maximize the overall profit. We assume this problem has been solved multiple times with different parameters

for each day or each decision point t . The framework is implemented on both fractional and binary knapsack problems. In the former, a portion of a product can be loaded in the container. On the contrary, only the whole product can be loaded in the latter. Therefore, the GOP is exactly the MILP introduced in (5.1) - (5.3) for the linear knapsack problem and by setting $\mathbf{x}^t \in \{0, 1\}$, the model for the binary knapsack problem is attained. For the linear knapsack problem, we compare the results of the linear regression model embedded in the framework with decision tree modeling. In contrast, in the discrete knapsack problem, we investigate using linear regression and classification tree models.

5.3.1.1 Data Generation

We adopted the data generation process presented in [26] while creating more complex sets of concealed operational rules. First, we construct an optimization model for each decision point t . The first step is to generate the parameters of the knapsack problems. We consider a small data set with $T = 500$ instances and $n = 15$ products and a larger one with $T = 500$ and $n = 50$. The parameters b and r_i are randomly generated from $[500, 1000]$ and $[50, 300]$, respectively, following a uniform distribution. We carry out our experiments considering different combinations of these parameters. a_i is also randomly sampled from a triangular distribution of $(8, b/n, b)$. In Addition, the big- M value is set to $\max(y_i) - \min(y_i)$. In the case of the knapsack problem, since $0 \leq y_i \leq 1$, the big- M equals 1.

According to Figure 5.1, the next step toward data generation is to solve the generated linear and binary knapsack problems. The obtained optimal solutions create the input features $(\bar{\mathbf{x}})$. To provide the output data, $\bar{\mathbf{y}}$, we generate an executed plan for each decision point by altering the optimal solution. To this end, several business rules are defined and applied to the optimal solution. The set of rules used in this work is shown in Table 5.1. Some of the settings consist of only one rule, while others combine several operational rules. The settings which are considered in Hewitt and Frejinger [26] are either similar to **Set1** with one rule or equivalent to **Set4** and **Set5** with several rules in the form of $(x_i \geq x_j)$. They also investigate a setting similar to **Set7**, where the corresponding business rule may vary depending on the day. Nonetheless, they choose between two rules randomly. Moreover, the authors do not consider testing on a setting similar to **Set2** and **Set3**, which are in place for all problem variables. As presented in the table, we attempt to evaluate the framework's performance on different types of conditional and non-conditional business rules.

In our experiments, we provide several rules that are close to business rules in industries. For instance, if the rules in **Set2** are applied in the case of the binary knapsack model, they restrict the loading of consecutive products together. As another example, consider the

Table 5.1 Rule settings for knapsack problem

Setting	Mathematical Representation of Rules	Setting	Mathematical Representation of Rules	Setting	Mathematical Representation of Rules
Set1	$x_1 \geq x_2$	Set2	if $x_i + x_{i+1} = 2 \rightarrow x_i = 0 \quad \forall i$	Set3	if $x_i + x_{i+1} + x_{i+2} \geq 2 \rightarrow x_i = 0 \quad \forall i$
Set4	$x_1 \geq x_2$ $x_{10} \geq x_{20}$ $x_{11} \geq x_{25}$ $x_{22} \geq x_{40}$ $x_{45} \geq x_{14}$ $x_{36} \geq x_{23}$	Set5	$x_1 \geq x_2, \quad x_{10} \geq x_{20}$ $x_{11} \geq x_{25}, \quad x_{22} \geq x_{40}$ $x_{45} \geq x_{14}, \quad x_{36} \geq x_{23}$ $x_{49} \geq x_{50}, \quad x_7 \geq x_{32}$ $x_{41} \geq x_{29}, \quad x_9 \geq x_{31}$	Set6	if $x_1 = 1 \ \& \ x_3 = 1 \ \& \ x_7 = 0 \rightarrow x_6 = 1$ if $x_1 = 0 \ \& \ x_2 = 1 \ \& \ x_5 = 1 \rightarrow x_3 = 0$ if $x_1 = 0 \ \& \ x_3 = 0 \rightarrow x_6 = 1$ if $x_4 = 1 \ \& \ x_7 = 0 \rightarrow x_9 = 0$ if $x_4 = 1 \ \& \ x_7 = 1 \rightarrow x_9 = 1$ if $x_1 = 1 \ \& \ x_2 = 1 \rightarrow x_3 = 0$
Set7	if $x_2 \leq 0.5 \rightarrow x_1 \geq x_2$ if $x_2 > 0.5 \rightarrow x_1 = 0$	Set8	if $x_2 > 0 \ \& \ x_1 = 0 \rightarrow x_3 = 0$ if $x_2 = 0 \ \& \ x_1 = 0 \rightarrow x_3 \geq x_4$	Set9	if $x_4 \leq 0.5 \rightarrow x_1 \geq x_2$ if $x_4 > 0.5 \rightarrow a_1x_1 + a_2x_2 + a_3x_3 \leq 0.4b$

second rule of **Set9**. It means that if the condition occurs, the total loaded portion of the first three products could not exceed 0.4 of the total container capacity. Therefore, the output $\bar{y}$ is generated by applying the rules on the optimal solutions. As an example, imagine **Set1** is imposed in a company. If the solution of the knapsack problem suggests $\bar{x}_1 = 0$ and $\bar{x}_2 = 0.5$, then the associated executed plan is considered as $\bar{y}_1 = \bar{y}_2 = 0.5$, even if the executed plan is not feasible for the GOP. In Table 5.2, we represent the functional mapping of some examples of the rules applied to the knapsack problem to elaborate on how we generate output data according to the obtained optimal solutions.

Table 5.2 Functional mapping of rules for knapsack problem

Mathematical Representation of Rule	Functional Mapping
$x_1 \geq x_2$	$\begin{cases} \bar{y}_i = \max[\bar{x}_1, \bar{x}_2] & i = 1 \\ \bar{y}_i = \bar{x}_i & i = 2, \dots, n \end{cases}$
$a_1x_1 + a_2x_2 + a_3x_3 \leq 0.4b$	$\begin{cases} \text{if } a_1\bar{x}_1 + a_2\bar{x}_2 + a_3\bar{x}_3 \leq 0.4b \rightarrow \bar{y}_i = \bar{x}_i & i = 1, \dots, n \\ \text{if } a_1\bar{x}_1 + a_2\bar{x}_2 + a_3\bar{x}_3 > 0.4b \rightarrow \bar{y}_i = \begin{cases} \min[\bar{x}_3, 0.4b/a_3] & i = 3 \\ \min[\bar{x}_2, (0.4b - a_3\bar{y}_3)/a_2] & i = 2 \\ \min[\bar{x}_1, (0.4b - a_3\bar{y}_3 - a_2\bar{y}_2)/a_1] & i = 1 \\ \bar{x}_i & i = 4, \dots, n \end{cases} \end{cases}$

5.3.1.2 Results

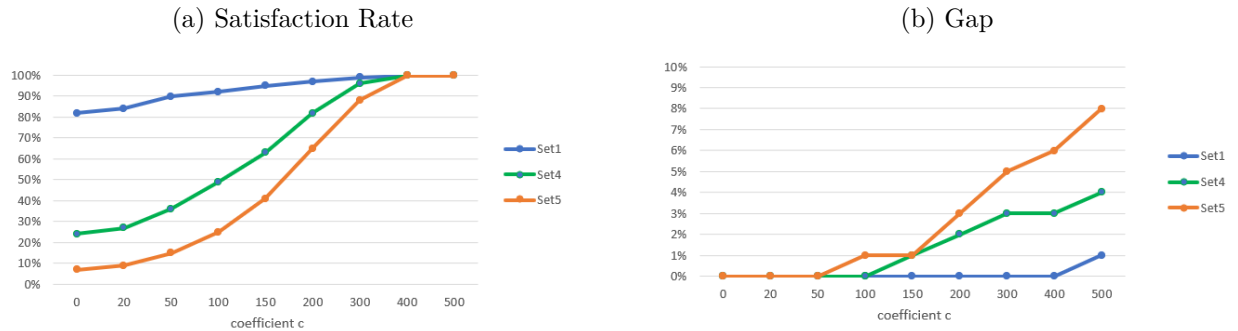
This section presents the framework's efficiency for linear and discrete knapsack problems under several rules defined in Table 5.1. All of the settings are investigated on the small data set with 15 variables, while **Set1**, **Set4**, and **Set5** are imposed on the larger data set with 50 variables. As mentioned in Section 5.2, there are two criteria to assess the quality of the obtained solutions by AOP. With the first measure, we determine the frequency of the instances in which the AOP provides a feasible solution regarding the hidden unmodeled constraints ($\mathbf{G}^t\mathbf{x} + \mathbf{H}^t\mathbf{v} \leq \mathbf{u}^t$) and we show it as a percentage of the whole data set. The next measure denotes the level of degradation in the objective function. We first need to

solve the GOP for each instance to compute this measure which we call the gap in the results in Table 5.3 and Table 5.4.

Next, we add the mathematical representation of the concealed rules as soft constraints. Then, we calculate the cost of the solutions obtained by the AOP and provide the percentage of the gap between these two objective functions by computing $(Z_{GOP-BR}^t - Z_{AOP}^t) / Z_{GOP-BR}^t$. In this ratio, Z_{GOP-BR}^t represents the objective value of the GOP where it contains $\mathbf{G}^t \mathbf{x} + \mathbf{H}^t \mathbf{v} \leq \mathbf{u}^t$, and Z_{AOP}^t is calculated as $\sum_{j=1}^n r_j x_j^{AOP}$ for each instance t . This measure is only considered for the instances which pass the feasibility test, therefore, we report an average gap of all the satisfied instances in the results.

When solving the AOP, we change the value of the penalty coefficient c in the objective function and evaluate the performance for each c . In most experiments, the satisfaction measure is boosted by increasing c , but it stabilizes beyond a certain value of c . However, the degradation in the objective value usually escalates by growing c . To elaborate more, consider Figure 5.2 in which we fit elastic Net regression on data of solving linear knapsack problem instances and then we employ the framework on our test set. We evaluate the framework for different values of c ranging from 0 to 500 for each of the three investigated sets of rules in the plots. They are all in the form of $x_i \leq x_j$, yet by different numbers of hidden operational rules. All the graphs in Figure 5.2 acknowledge that the satisfaction rate augments by increasing the value of c . In addition, we see that for all numbers of hidden business rules, the solutions of the AOP are far more likely to satisfy the rules compared to the GOP ($c = 0$). Although this satisfaction converges to 100% in all three sets of rules, according to Figure 5.2, the gap is normally surged. Therefore, the best coefficient for **Set1** is 300 with no degradation in the objective value, while the best results are obtained for **Set4** and **Set5** in $c = 400$ with a gap equal to 3% and 6%, respectively. In general, comparing the graphs demonstrates that by increasing the number of rules, the quality measures decrease.

Figure 5.2 Effect of coefficient c on the quality of the solutions of eNet linear regression on the linear knapsack problem.



We demonstrate the efficiency metrics for six settings tested on the linear knapsack problem as well as six settings on the binary knapsack problem in Tables 5.3 and 5.4, respectively, considering that the reported results are associated to the test instances. Remark that we only report the best results in terms of feasibility obtained by altering c and the associated gap. We highlight the best-obtained satisfaction rate among all the methods in the tables as the more important measure; hence, if a tie occurs, we also highlight the best gap.

The first column of the results denoted in both tables of this section is related to the percentage of instances that are already feasible in the GOP regarding the concealed constraints ($c = 0$). In the next columns of the tables, we compare the performance of linear regression, lasso regression, ridge regression, and elastic Net regression on both linear and binary knapsack problems. The last column of each learning method belongs to the sparsity of the coefficient matrix, while we report an average of the sparsity for the altered variables of each set of rules. Furthermore, we investigate applying regression trees on the linear knapsack problem and classification trees on the 0-1 knapsack problem.

Table 5.3 Results of applying the framework on different settings for the linear knapsack problem

Setting	Satisfaction $c = 0$	Linear Regression			Lasso Linear Regression			Ridge Linear Regression			eNet Linear Regression			Regression Tree	
		Satisfaction	Gap	Sparsity	Satisfaction	Gap	Sparsity	Satisfaction	Gap	Sparsity	Satisfaction	Gap	Sparsity	Satisfaction	Gap
Set1	81%	92%	0%	0%	100%	0%	74%	93%	0%	0%	100%	0%	80%	82%	0%
Set4	24%	54%	3%	0%	100%	3%	88%	62%	4%	0%	100%	3%	93%	24%	0%
Set5	7%	27%	10%	0%	100%	6%	88%	40%	7%	0%	100%	6%	91%	7%	0%
Set7	85%	85%	0%	0%	85%	0%	40%	85%	0%	0%	85%	0%	87%	86%	0%
Set8	75%	86%	4%	0%	82%	3%	67%	86%	4%	0%	82%	3%	67%	80%	0%
Set9	75%	97%	3%	0%	97%	2%	76%	97%	2%	0%	97%	2%	73%	80%	0%

Based on the results in Table 5.3, we see that in most of the experiments applying the framework enhances the feasibility level compared to the solutions of GOP ($c = 0$). Next, we compare the quality of the solutions regarding the objective function. It shows that overall, solving the AOP obtained by different methods yields a solution with objective function values which are very close to the GOP values (the maximum average difference is 10%). The table suggests that the performance of lasso and eNet are similar, and, for the first three settings, they satisfy the constraints for all the instances. We observe that even simple linear regression without any regularization boosts the feasibility rate. The only setting which is not significantly improved through the framework is **Set7**. One hypothesis is that since the regression score is not as good as the other settings, the regression equations cannot represent the constraints well. However, investigating all the settings with different machine learning methods, we could not concretely conclude that there is always a significant, meaningful relation between the learning score and the framework's performance. Another possible effective attribute is the sparsity of the coefficient matrix. Looking into both Tables

5.3 and 5.4 depicts that in most cases, regression methods with a sparser coefficient matrix outperform those employing a denser coefficient matrix. As an example, consider **Set4**, in which, although the learning scores are very close to each other, $r^2 = 0.85$ for linear regression and $r^2 = 0.86$ for eNet, the sparsity of the coefficient matrix in eNet results in a better quality of solutions in terms of satisfaction rate.

Table 5.4 Results of applying the framework on different settings for 0-1 knapsack problem

Setting	Satisfaction $c = 0$	Linear Regression			Lasso Linear Regression			Ridge Linear regression			eNet Linear Regression			Classification Tree	
		Satisfaction	Gap	Sparsity	Satisfaction	Gap	Sparsity	Satisfaction	Gap	Sparsity	Satisfaction	Gap	Sparsity	Satisfaction	Gap
Set2	23%	100%	15%	0%	100%	12%	62%	100%	20%	0%	100%	8%	69%	100%	0%
Set3	6%	100%	30%	0%	100%	29%	58%	100%	30%	0%	100%	29%	70%	100%	19%
Set6	52%	96%	29%	0%	98%	24%	42%	98%	29%	0%	98%	27%	38%	100%	12%
Set7	90%	100%	1%	0%	100%	1%	67%	100%	1%	0%	100%	1%	67%	100%	0%
Set8	80%	100%	5%	0%	100%	3%	33%	100%	5%	0%	100%	3%	53%	100%	3%
Set9	77%	97%	3%	0%	97%	3%	71%	97%	3%	0%	97%	3%	44%	99%	0%

The next set of experiments belongs to the binary knapsack problem depicted in Table 5.4. One notable observation is that the table confirms the superiority of the framework for the binary knapsack problem as a discrete problem. Particularly, we see that regardless of the type of machine learning method, most of the solutions are satisfied after employing the framework. The AOP could pass the feasibility test for all the instances for four of the six tested settings. It is the case even for **Set2** and **Set3**, with much less frequency of feasibility for the GOP solutions ($c = 0$). We next turn to more detailed comparisons between the methods. Evidently, although lasso and eNet regressions still outperform ridge and linear regressions, the classification tree model suggests the best results in terms of both satisfaction and gap. Specifically, for the first three sets of operational rules, which impact the objective function of the problem hugely. We see that the average gap obtained by the classification tree is very competitive. At the same time, the solutions satisfy the hidden constraints for all of the test instances.

Evaluating both tables simultaneously and comparing quality measures for all the methods together confirm the superiority of eNet and lasso linear regression over the other regression methods. The coefficient matrices are always very sparse in these models. In addition, clearly, the classification tree model is the most efficient method for 0-1 knapsack problem instances.

5.3.2 Nurse Rostering Problem

We investigate the framework's performance on a more practical and complex optimization problem. Nurse rostering [121] is an essential aspect of healthcare management dealing with scheduling nurses with different skills and has received considerable attention in recent years. Nurse rostering involves assigning shifts to nurses for a time horizon while restricting them

to a set of hard and soft constraints. The main objective is to optimize the utilization of limited resources, thereby enhancing the operational efficiency of the hospital. However, it is important to prioritize the job satisfaction of nurses in the process. The Nurse Rostering Problem is a complex combinatorial optimization problem and is known to be NP-hard [121]. Generally, hospitals dedicate considerable effort to creating high-quality rosters for their nurses, including using decision-support tools and expert judgments.

In modeling the NRP, the constraints taken into account and the objectives could vary from hospital to hospital, or in different planning periods within a hospital. This diversity leads to a wide range of nurse rostering optimization models. Moreover, the solution methodologies proposed in the literature for this problem originate in different categories of MILP, CP, heuristics, meta-heuristics, stochastic programming, etc. We refer the reader to Ngoo et al. [122] and the references therein for more details.

In this section, we introduce a basic NRP formulated and solved using MILP. We then examine utilizing the proposed data-driven methodology to tackle this problem. Note that, this study represents an initial attempt in this field, and our objective is to gauge the efficacy of the methodology when confronted with a limited amount of training data. Hence, we opt for restricting our analysis to scenarios where all nurses are identical in terms of their skills and preferences. This decision allows us to provide additional training data from the available dataset, and we outline the details of how this is achieved later in this section.

Inspired by the models presented in [123, 124, 125], a MILP formulation of the problem is given below:

Parameters

N	Set of nurses: $\{1, \dots, n\}$.
P	Set of days in the planning horizon: $\{1, \dots, p\}$.
S	Set of shift types: $\{1, \dots, s\} = (\text{morning, evening, over-night})$.
D_{jd}	Nurse demand for shift j in day d .
α	Penalty for understaffing.
α'	Penalty for overstaffing.
K_1	The minimum number of shifts that each nurse must be assigned.
K_2	The maximum number of shifts that each nurse can be assigned.
K_3	The maximum number of consecutive working days each nurse can work.
K_4	The minimum number of free days each nurse must have after 2 consecutive overnight shift.
K_5	The maximum number of overnight shifts each nurse can be assigned.

Decision Variables

$x_{ijd} \in \{0, 1\}$	1 if nurse i assigned to shift j on day d , 0 otherwise.
$w_{jd} \geq 0$	Total understaffing for shift j on day d .
$w'_{jd} \geq 0$	Total overstaffing for shift j on day d .

The objective function is to minimize the total under and overstaffing:

$$\min \sum_{j=1}^s \sum_{d=1}^m (\alpha w_{jd} + \alpha' w'_{jd}) \quad (5.25)$$

Hence, the covering constraints to demonstrate the understaffing and overstaffing are given:

$$w_{jd} \geq D_{jd} - \sum_{i \in N} x_{ijd} \quad \forall d \in P, \quad \forall j \in S \quad (5.26)$$

$$w'_{jd} \geq \sum_{i \in N} x_{ijd} - D_{jd} \quad \forall d \in P, \quad \forall j \in S \quad (5.27)$$

The hard constraints of the problem which have to be satisfied are as follows:

- **C1:** Any nurse is allowed to work no more than one shift per day.

$$\sum_{j=1}^s x_{ijd} \leq 1 \quad \forall i \in N, \quad \forall d \in P \quad (5.28)$$

- **C2:** The total number of shifts assigned to each nurse must be within a minimum and a maximum number.

$$K_1 \leq \sum_{j=1}^s \sum_{d=1}^p x_{ijd} \leq K_2 \quad \forall i \in N \quad (5.29)$$

- **C3:** The number of consecutive working days assigned to a nurse must not exceed the maximum allowed.

$$\sum_{j=1}^s \sum_{d=k}^{k+K_3} x_{ijd} \leq K_3 \quad \forall i \in N, \quad \forall k : 1, \dots, (p - K_3) \quad (5.30)$$

- **C4:** A nurse has to have a minimum number of free days after 2 consecutive night shifts.

$$\sum_{d=k}^{k+1} x_{i3d} + \sum_{j=1}^s x_{ij(k+1+r)} \leq 2 \quad \forall i \in N, \quad \forall k : 1, \dots, (p - K_4 - 1), \quad \forall r : 1, \dots, K_4 \quad (5.31)$$

- **C5:** A nurse cannot work more than a maximum number of night shifts during the planning horizon.

$$\sum_{d=1}^p x_{i3d} \leq K_5 \quad \forall i \in N \quad (5.32)$$

This is the basic model we specify as our GOP of nurse rostering. In the next subsection, we clarify the business rules we consider for testing and how we generate input and output for the machine learning models.

5.3.2.1 Data generation

To generate our historical data, similar to the procedure that has been carried out for the knapsack problem, we need to solve the NRP for each planning period t . Without loss of generality, we assume the regulations and demand can vary from time to time in a hospital. Therefore, for each planning period, we randomly generate our model's parameters $D_{j,d}$, K_1 , K_2 , K_3 , K_4 and K_5 from uniform distributions of $[0, 3]$, $[4, 8]$, $[8, 9]$, $[5, 6]$, $[1, 2]$ and $[2, 5]$, respectively. Note that we implement and report two separate sets of tests, wherein one of them, we consider penalty coefficients α and α' equal to 1, meaning that the hospital attempts to avoid understaffing and overstaffing with the same weight. While in the other set of tests, replacing α by 3 and α' by 1, we assess the scenario in which understaffing is more unfavorable for the hospital than overstaffing.

As discussed in Section 5.3.2, in the current study, we assume the nurses are identical so that they can be replaced by each other, and therefore, the solutions of the NRP are symmetric. This assumption generates more training data from an initial set of solved optimization instances. All generated data sets consist of 200 initial distinct planning periods divided into 80% for training and 20% for testing. However, after solving this set of instances, we use the optimal solutions to provide more training data by considering every schedule for each nurse as a training instance. Thus, the new set of instances is multiplied by the number of nurses.

Table 5.5 Rule setting for Nurse Rostering Problem

Setting	Description of Rule	Logical/Mathematical Representation of Rule
Set1	After a night shift on, the following morning shift is off	if $x_{i3d} = 1 \rightarrow x_{i1(d+1)} = 0 \quad \forall i, \quad \forall d : 1, \dots, p-1$
Set2	After a night shift on the following morning and afternoon shifts are off	if $x_{i3d} = 1 \rightarrow x_{i1(d+1)} + x_{i2(d+1)} = 0 \quad \forall i, \quad \forall d : 1, \dots, p-1$
Set3	After a shift in a week-end on, the two first week-days are off	if $\sum_j (x_{i,j,7q-1} + x_{i,j,7q}) \geq 1 \rightarrow \sum_j (x_{i,j,7q+1} + x_{i,j,7q+2}) = 0 \quad \forall i, \quad \forall q : 1, \dots, \lfloor p/7 \rfloor$

By solving the generated NRP instances, we acquire the input features of our learning models. Hence, following the same steps as the knapsack problem, we alter the optimal solution based on some assumed operational rules to generate output features. In Table 5.5, we display three different business rules that are not included in the GOP, and executable plans should satisfy them. To indicate the usability of the proposed framework in a real industrial problem, we devise some business rules that are practical and interpretable in the NRP [126, 125]. In

addition to the logical representation of the rules depicted in this table, the constraints which are added to the MILP model are described in a related table in Appendix C.1. We should remark that where we test the rules associated with **Set1** and **Set2**, we solve the NRP model of (5.25)-(5.32); nonetheless, in order to test **Set3**, to deal with a more complicated problem, we consider **Set2** as a part of constraints in the GOP model of NRP in addition to constraints (C1-C5).

Since the NRP problem's principal decision variable, x_{ijd} , has three indices, the regression and branching equations in the corresponding AOP slightly differ from the knapsack problem. As an instance, the equation 5.16 is replaced by the following:

$$y_{ijd} - (\hat{\beta}^{ijdl} + \sum_{efg} (\beta_{efg}^{ijdl} x_{efg})) \leq M(1 - z^{ijdl}) \quad \forall i, j, d \quad \forall l \in L \quad (5.33)$$

And in the case of branching instead of 5.18 in the AOP, we embed the following equation:

$$\sum_{efg} (h_{efg}^{ijdm} x_{efg}) \leq g^{ijdm} + (1 - z^{ijdl}) \quad \forall i, j, d \quad \forall l \in leaf, \quad \forall m \in Left(l) \quad (5.34)$$

We also need to learn the models based on two indices, j and d , since the nurses are interchangeable. Nevertheless, the equations in the AOP must be compatible with the true decision variable of the problem x_{ijd} . Therefore, the equations (5.33) and (5.34) are reformulated as below, respectively:

$$y_{ijd} - (\hat{\beta}^{jdl} + \sum_{fg} (\beta_{fg}^{jdl} x_{ifg})) \leq M(1 - z^{jdl}) \quad \forall i, j, d \quad \forall l \in L \quad (5.35)$$

$$\sum_{fg} (h_{fg}^{jdm} x_{ifg}) \leq g^{jdm} + (1 - z^{jdl}) \quad \forall i, j, d \quad \forall l \in leaf, \quad \forall m \in Left(l) \quad (5.36)$$

5.3.2.2 Results

To showcase the framework's performance for the Nurse Rostering Problem, we provide information on four types of gaps and the percentage of rule satisfaction for the test instances. The gaps are defined in Table 5.6. The first computed gap is similar to the gap we previously reported for the knapsack problems. It illustrates the degradation in the objective function of the problem and is expressed as a percentage of the AOP objective value. The second type of gap measures the difference between the AOP and the GOP solutions with known constraints in terms of the total deviation from the demand. The understaffing and overstaffing gaps are included to provide information on the variations between the AOP and the GOP solutions with additional known constraints regarding nurse shortage and over-coverage during the

planning period. We use the absolute value for overstaffing and understaffing gaps, and all four reported gaps are the average of the gaps of all test instances.

Table 5.6 Definition of computed gaps for Nurse Rostering Problem

Different Computed Gaps	Definition	Computation
Gap1	Objective Function Degradation	$(Z_{AOP-BR}^t - Z_{GOP}^t) / Z_{AOP-BR}^t$
Gap2	Total Demand Deviation Gap	$\sum_{jd}(w + w')^{AOP} - \sum_{jd}(w + w')^{GOP}$
Gap3	Understaffing Gap	$\sum_{jd}(w)^{AOP} - (w)^{GOP}$
Gap4	Overstaffing Gap	$\sum_{jd}(w')^{AOP} - (w')^{GOP}$

To compare the quality of solutions obtained by the AOP modeled by linear regression and the AOP modeled by classification tree, we provide two tables for the NRP. Table 5.7 is associated with the Nurse Rostering Problem (NRP), where avoiding understaffing is more critical for the decision-makers in the hospital. In contrast, Table 5.8 displays the results where understaffing and overstaffing have equal weight. For both types of experiments, we test the framework on various rule settings and problem sizes, where n is the number of nurses and p corresponds with the planning horizon. Like the knapsack problem experiments, we tune the coefficient c in the process. In this problem, we vary c from 0.1 to 15, yet only report the best results.

To ensure meaningful and comparable results, we include a column in the tables that indicates the average nurse demand for the planning period in the test set. This enables us to compare the gaps to the demands for each test and draw conclusions based on the findings.

One remark that should be considered is that solving the NRP using the framework with an embedded classification tree is very fast, taking less than 20 seconds for each instance. However, we have noticed that solving some instances using linear regression equations to optimality is very time-consuming. As a result, in our experiments, we limit the time for solving each AOP model to 600 seconds.

Looking into the satisfaction rates presented in Tables 5.7 and 5.8, the AOP with both linear regression models and classification trees is feasible regarding all instances' operational rules. This satisfaction is obtained when we increase the coefficient c to some extent. We next turn our attention to the various reported gaps in the tables. We observe that the classification tree consistently outperforms the embedded linear regression models regarding all gap types for all test settings. A closer look into the result tables reveals that even for the settings where the objective function degradation for linear regression models exceeds 70%, the classification tree achieves the 0% gap. This is because we need to increase the coefficient c significantly for embedded linear regression models to align the optimal solution with the prediction of the executable plan and satisfy the hidden rules. In contrast, embedding the obtained classification tree in the framework allows us to achieve feasible solutions regarding

the operational rules with a small value of c (e.g., $c = 0.1$). Although the optimal solution obtained by the GOP with known rules and the AOP based on classification trees may vary, the demand covering, and consequently, the objective value remains almost the same for all instances.

Table 5.7 Results of applying the framework on different settings for Nurse Rostering Problem – $\alpha = 3, \alpha' = 1$

Setting	Problem Size	Satisfaction $c = 0$	Demand	Linear Regression						eNet Linear Regression						Classification Tree				
				Satisfaction	Gap1	Gap2	Gap3	Gap4	Sparsity	Satisfaction	Gap1	Gap2	Gap3	Gap4	Sparsity	Satisfaction	Gap1	Gap2	Gap3	Gap4
Set1	$n = 6 \ p=10$	62%	44.2	100%	30%	4.30	3.18	1.12	0%	100%	2%	0.12	0.10	0.02	43%	100%	0%	0	0	0
Set2	$n = 6 \ p=10$	27%	44.2	100%	60%	10.25	6.30	3.95	0%	100%	56%	8.62	6.05	2.57	49%	100%	0%	0	0	0
Set2	$n = 7 \ p=12$	15%	54.1	100%	76%	21.57	13.57	8.00	0%	100%	74%	18.70	12.82	5.87	46%	100%	0%	0	0	0
Set3	$n = 7 \ p=12$	0%	54.1	100%	42%	8.12	5.00	3.12	0%	100%	41%	7.67	4.80	2.87	57%	100%	1%	0.10	0.08	0.02

Here, we only demonstrate the performance of linear regression without regularization as well as eNet, as they represent the least and most effective approaches, respectively. We confirm the superiority of eNet over linear regression without regularization for the NRP in addition to the knapsack problem.

Considering both tables, it can be deduced that although adding more complicated business rules and increasing the size of the problem results in less rule satisfaction in the GOPs, embedding the machine learning models still remains efficient in satisfying the hidden rules. More significantly, using the framework with the classification tree is highly effective even in the last setting with a larger instance size regarding satisfaction and all types of gaps.

Table 5.8 Results of applying the framework on different settings for Nurse Rostering Problem – $\alpha = \alpha' = 1$

Setting	Problem Size	Satisfaction $c = 0$	Demand	Linear Regression						eNet Linear Regression						Classification Tree				
				Satisfaction	Gap1	Gap2	Gap3	Gap4	Sparsity	Satisfaction	Gap1	Gap2	Gap3	Gap4	Sparsity	Satisfaction	Gap1	Gap2	Gap3	Gap4
Set1	$n = 6 \ p=10$	50%	44.2	100%	46%	5.25	2.97	2.27	0%	100%	44%	4.62	4.17	0.45	43%	100%	0%	0	0	0
Set2	$n = 6 \ p=10$	30%	44.2	100%	72%	19.05	11.70	7.35	0%	100%	62%	10.90	8.10	2.80	50%	100%	0%	0	0	0
Set2	$n = 7 \ p=12$	7%	54.1	100%	79%	24.82	16.37	8.45	0%	100%	78%	23.72	16.02	7.70	48%	100%	0%	0	0	0
Set3	$n = 7 \ p=12$	0%	54.1	100%	40%	7.10	4.65	2.45	0%	100%	38%	6.75	4.40	2.35	56%	100%	1%	0.10	0.08	0.02

5.4 Conclusion

In this study, we extend a data-driven framework that was recently proposed to learn and incorporate customization constraints of the classical linear programming models using linear regression and statistical learning. We incorporate more complex machine learning models (lasso, ridge, eNet regularization and decision tree) into the framework, evaluate their performance on more complex discrete models and test conditional rules.

Furthermore, we empirically validate the method on discrete problems for the first time. We investigate the framework's performance on the binary knapsack problem and a more

realistic and complex problem in the field of nurse rostering, where operational rules are more practical and interpretable.

The notable outcome of the study is that using different types of regularized linear regression enhances the results in most instances of all problems compared to linear regression without regularization. Moreover, we conclude that embedding classification trees in binary problems outperforms embedding regression models to a great extent.

Finally, this study is still at an exploratory level. As an example, while learning linear models and embedding them in the framework is less computationally expensive, investigating the application of other machine learning methods like neural networks is of interest [127]. Moreover, a possible future research direction is learning and implementing separating hyperplanes [116] in the framework for more complex rules.

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS

In this dissertation, we studied two applications of data-driven methods at different levels to tackle large-scale optimization problems. One level is related to the underlying structure of the problem and problem's type. The other is the instance level in which extracting patterns from data instances is essential. In this chapter, we highlight the contributions achieved and conclude the dissertation. Then, we discuss the limitations of this study and possible future research directions.

6.1 Summary of Works

According to the literature review of Chapter 2, it is clear that there are insufficient studies on investigating the structure of the Binary Quadratic Programming problems and exploiting patterns to tackle solving those. There are only a few papers dedicated to reformulating a specific BQP problem based on its corresponding structure and employing CG to solve it. However, there is not a unified framework to cover multiple problems in this class of Combinatorial Optimization problems. Moreover, looking into the literature on data-driven methods for customizing the constraints of classical optimization models, we notice that there is only one study that leverages data-driven methods to customize constraints of MILP according to the requirements of each company.

In the first and second studies, we investigate the structure level of the optimization models to provide a framework for solving the Binary Quadratic Programming problems. More specifically, in Chapter 3, we examine a star-based framework proposed for solving AQMSTP on other problems of the adjacent-only class of BQPs. We represent how this reformulation introduced for the specific problem in the literature can be generalized for other problems. Based on this study, in Chapter 4, we propose a unified framework to reformulate a BQP problem with linear constraints to a new BQP with an exponential number of variables defined on a graph. The framework relies on the concept of stars in the graph to split the quadratic costs into adjacent and non-adjacent components indicating in-star and out-of-star interactions. This study is among the very few studies representing a reformulation and solution method by exploiting the structure of the problem for the BQP problems. We attempt to answer this fundamental question: for which type of structure of the quadratic matrix and to some extent using the star-reformulation is beneficial? The frameworks were tested on different instances of the AQSAP, MOT and QSAP. Computational results indicate the superiority of the framework in terms of both computational times and LB over

one of the state-of-the-art solvers, GUROBI, when applied to various benchmark reformulations of problems with adjacent-only quadratic costs. When it comes to testing BQPs with both adjacent and non-adjacent costs, the performance of the method decreases; however, even a basic implementation of the framework still depicts promising results, specifically for instances with sparser quadratic matrices.

As for the third essay of this dissertation explained in Chapter 5, we investigate a data-driven method at the instance level, meaning that we utilize such methods for exploring patterns in data of an optimization problem and tackling other instances of the problem. Optimization software is used by decision-makers in different industries. In numerous practical applications, these optimization tools are often not readily adjustable or configurable by end users due to limited knowledge, resources, or the required investment to consistently make such customizations. Planners often manually adjust software solutions based on implicit internal operational rules and preferences to make them practical. In this study, we develop a data-driven prediction model to capture the underlying rules that decision-makers take into account when altering the optimal solutions. Based on the previous studies in the literature, our main contribution here is to leverage regularized linear regression models and decision trees as predictors and embed them as extra constraints in the MILP model of the industry. In addition, a challenging NRP is considered as a case study of the 0-1 MILP and we depict how to embed a classification tree instead of linear regression to tackle this class of problems. The results demonstrate a huge improvement in the performance measures using the proposed framework compared to the previous study in this area.

6.2 Limitations and Future Research

In the first and second studies have been described in Chapter 3 and Chapter 4, respectively, we provide reformulations for BQPs and we use a decomposition to solve these problems. Although two adjacent-only problems and one problem with both adjacent and non-adjacent costs are tested, more experiments on other quadratic problems with diverse constraint structures are of interest. As an example, exploring the cost-splitting technique on general BQP problems such as the general case of the Quadratic Minimum Spanning Tree Problem is another avenue for further research. Alternatively, exploring the star-reformulation idea on more problems in adjacent-only BQP class, for instance, on the adjacent Quadratic Assignment Problem is another research direction for future studies. Testing more accelerating techniques such as stabilization methods and dual optimal inequalities for the CG can potentially increase the efficiency of the framework. A future research direction is to explore the possibility of incorporating the proposed framework in a branch-and-bound tree to improve

the primal bounds in addition to the dual bounds. As we discussed before, compared to MILP, there are not enough studies on using data-driven methods for BQPs. Data-driven methods that are proposed in the literature for linear programming might be adapted to tackle the BQP problems as well.

In the third essay represented in Chapter 5, which discusses customizing the constraints of classical optimization problems by finding patterns in historical data of industry, the major limitation has been the lack of access to real learning data. In this study, we generate random data for both the knapsack problem and the NRP as the input features and alter them based on some predetermined rules as output features. It would be much more desirable to test the performance of the framework on some real data and executed plans of an industrial application. Another limitation of the study is related to manually finding a promising penalty coefficient in the objective function of the optimization model. One possible direction for future research is improving the framework to tune this hyper-parameter automatically. We also believe that it is worth to try embedding some non-linear machine learning models in the framework and observing the performance in terms of satisfaction, gap, and computational time.

REFERENCES

- [1] Korte Bernhard and Jens Vygen. Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*, 2008.
- [2] Richard M Karp. Reducibility among combinatorial problems, complexity of computer computations (re miller and jw thatcher, editors), 1972.
- [3] Clarisse Dhaenens and Laetitia Jourdan. *Metaheuristics for big data*. John Wiley & Sons, 2016.
- [4] Roberto Henschel, Laura Leal-Taixé, Daniel Cremers, and Bodo Rosenhahn. Fusion of head and full-body detectors for multi-object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1428–1437, 2018.
- [5] Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, and Renato Umeton. *Machine Learning, Optimization, and Big Data: Third International Conference, MOD 2017, Volterra, Italy, September 14–17, 2017, Revised Selected Papers*, volume 10710. Springer, 2017.
- [6] Taghi Khaniyev. Data-driven structure detection in optimization: Decomposition, hub location, and brain connectivity. 2018.
- [7] Stephen Boyd, Lin Xiao, Almir Mutapcic, and Jacob Mattingley. Notes on decomposition methods. *Notes for EE364B, Stanford University*, 635:1–36, 2007.
- [8] Abraham P Punnen, Pooja Pandey, and Michael Friesen. Representations of quadratic combinatorial optimization problems: A case study using quadratic set covering and quadratic knapsack problems. *Computers & Operations Research*, 112:104769, 2019.
- [9] Eranda Çela. *The quadratic assignment problem: theory and algorithms*, volume 1. Springer Science & Business Media, 2013.
- [10] David Pisinger. The quadratic knapsack problem—a survey. *Discrete applied mathematics*, 155(5):623–648, 2007.
- [11] Anja Fischer. An analysis of the asymmetric quadratic traveling salesman polytope. *SIAM Journal on Discrete Mathematics*, 28(1):240–276, 2014.

- [12] Borzou Rostami, Federico Malucelli, Pietro Belotti, and Stefano Gualandi. Lower bounding procedure for the asymmetric quadratic traveling salesman problem. *European Journal of Operational Research*, 253(3):584–592, 2016.
- [13] Abraham P Punnen, Matthias Walter, and Brad D Woods. A characterization of linearizable instances of the quadratic traveling salesman problem. *arXiv preprint arXiv:1708.07217*, 2017.
- [14] Hao Hu and Renata Sotirov. Special cases of the quadratic shortest path problem. *Journal of Combinatorial Optimization*, 35(3):754–777, 2018.
- [15] Borzou Rostami, André Chassein, Michael Hopf, Davide Frey, Christoph Buchheim, Federico Malucelli, and Marc Goerigk. The quadratic shortest path problem: complexity, approximability, and solution methods. *European Journal of Operational Research*, 268(2):473–485, 2018.
- [16] Arjang Assad and Weixuan Xu. The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)*, 39(3):399–417, 1992.
- [17] Borzou Rostami and Federico Malucelli. Lower bounds for the quadratic minimum spanning tree problem based on reduced cost computation. *Computers & Operations Research*, 64:178 – 188, 2015. ISSN 0305-0548. doi: <https://doi.org/10.1016/j.cor.2015.06.005>. URL <http://www.sciencedirect.com/science/article/pii/S0305054815001501>.
- [18] Dilson Lucas Pereira and Alexandre Salles da Cunha. Dynamic intersection of multiple implicit dantzig–wolfe decompositions applied to the adjacent only quadratic minimum spanning tree problem. *European Journal of Operational Research*, 284(2):413–426, 2020.
- [19] Bruno Escoffier and Peter L Hammer. Approximation of the quadratic set covering problem. *Discrete Optimization*, 4(3-4):378–386, 2007.
- [20] Dilson Lucas Pereira, Michel Gendreau, and Alexandre Salles da Cunha. Stronger lower bounds for the quadratic minimum spanning tree problem with adjacency costs. *Electronic Notes in Discrete Mathematics*, 41:229–236, 2013.
- [21] Frank Fischer, Gerold Jaeger, Anja Lau, and Paul Molitor. Complexity and algorithms for the traveling salesman problem and the assignment problem of second order. *Lecture Notes in Comput. Sci*, 5165:211–224, 11 2009.

- [22] Borzou Rostami, Federico Malucelli, Davide Frey, and Christoph Buchheim. On the quadratic shortest path problem. In Evripidis Bampis, editor, *Experimental Algorithms*, pages 379–390. Springer International Publishing, Cham, 2015. ISBN 978-3-319-20086-6.
- [23] Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, and Tao Jiang. On the approximation of correlation clustering and consensus clustering. *Journal of Computer and System Sciences*, 74(5):671–696, 2008.
- [24] Daniel Aloise, Sonia Cafieri, Gilles Caporossi, Pierre Hansen, Sylvain Perron, and Leo Liberti. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.
- [25] Julian Yarkony, Yossiri Adulyasak, Maneesh Singh, and Guy Desaulniers. Data association via set packing for computer vision applications. *Inform Journal on Optimization*, 2(3):167–191, 2020.
- [26] Mike Hewitt and Emma Frejinger. Data-driven optimization model customization. *European Journal of Operational Research*, 287(2):438–451, 2020.
- [27] João Vilela, Bruno Fanzeres, Rafael Martinelli, and Claudio Contardo. Efficient labeling algorithms for adjacent quadratic shortest paths. *arXiv preprint arXiv:2112.04045*, 2021.
- [28] Borzou Rostami, Federico Malucelli, and Pietro Belotti. A tight lower bound for the adjacent quadratic assignment problem. 2014.
- [29] Dilson Lucas Pereira, Michel Gendreau, and Alexandre Salles da Cunha. Branch-and-cut and branch-and-cut-and-price algorithms for the adjacent only quadratic minimum spanning tree problem. *Networks*, 65(4):367–379, 2015.
- [30] Dilson Lucas Pereira and Alexandre Salles da Cunha. Polyhedral results, branch-and-cut and Lagrangian relaxation algorithms for the adjacent only quadratic minimum spanning tree problem. *Networks*, 71(1):31–50, 2018.
- [31] Francisco Barahona. The max-cut problem on graphs not contractible to K5. *Operations Research Letters*, 2(3):107–111, 1983.
- [32] Warren P Adams and Hanif D Sherali. Linearization strategies for a class of zero-one mixed integer programming problems. *Operations Research*, 38(2):217–226, 1990.

- [33] Warren P. Adams and Richard J. Forrester. A simple recipe for concise mixed 0-1 linearizations. *Operations Research Letters*, 33(1):55 – 61, 2005. ISSN 0167-6377. URL <http://www.sciencedirect.com/science/article/pii/S0167637704000562>.
- [34] Hanif D Sherali and J Cole Smith. An improved linearization strategy for zero-one quadratic programming problems. *Optimization Letters*, 1(1):33–47, 2007.
- [35] Leo Liberti. Compact linearization for binary quadratic problems. *4OR*, 5(3):231–245, 2007.
- [36] Sven Mallach. Compact linearization for binary quadratic problems subject to assignment constraints. *4OR*, 16:295–309, 2018.
- [37] Peter M Hahn, Yi-Rong Zhu, Monique Guignard, William L Hightower, and Matthew J Saltzman. A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. *INFORMS Journal on Computing*, 24(2):202–209, 2012.
- [38] Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.
- [39] Warren P Adams and Hanif D Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.
- [40] Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.
- [41] Geraldo Regis Mauri and Luiz Antonio Nogueira Lorena. Lagrangean decompositions for the unconstrained binary quadratic programming problem. *International Transactions in Operational Research*, 18(2):257–270, 2011.
- [42] Geraldo Regis Mauri and Luiz Antonio Nogueira Lorena. A column generation approach for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, 217(1):69 – 74, 2012. ISSN 0377-2217. doi: <http://dx.doi.org/10.1016/j.ejor.2011.09.016>. URL <http://www.sciencedirect.com/science/article/pii/S0377221711008198>.

- [43] Wei-An Chen, Zhen Zhu, and Nan Kong. A Lagrangian decomposition approach to computing feasible solutions for quadratic binary programs. *Optimization Letters*, Mar 2017. ISSN 1862-4480. doi: 10.1007/s11590-017-1125-x. URL <https://doi.org/10.1007/s11590-017-1125-x>.
- [44] Alain Billionnet and Éric Soutif. An exact method based on Lagrangian decomposition for the 0–1 quadratic knapsack problem. *European Journal of Operational Research*, 157(3):565–575, 2004.
- [45] Pierre De La Poix De Fréminville, Guy Desaulniers, Louis-Martin Rousseau, and Sylvain Perron. A column generation heuristic for districting the price of a financial product. *Journal of the Operational Research Society*, 66(6):965–978, 2015.
- [46] C. Helmberg, F. Rendl, and R. Weismantel. A semidefinite programming approach to the Quadratic Knapsack Problem. *Journal of Combinatorial Optimization*, 4(2):197–215, 2000. ISSN 1382-6905. doi: 10.1023/A:1009898604624. URL <http://dx.doi.org/10.1023/A%3A1009898604624>.
- [47] Claude Lemarechal Francois Oustry. SDP relaxations in combinatorial optimization from a Lagrangian viewpoint. *Advances in Convex Analysis and Global Optimization: Honoring the Memory of C. Caratheodory (1873-1950)*, 54:119–134, 2001.
- [48] Alain Billionnet, Sourour Elloumi, and Marie-Christine Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The qcr method. *Discrete Applied Mathematics*, 157(6):1185–1197, 2009.
- [49] Borzou Rostami, Fausto Errico, and Andrea Lodi. A convex reformulation and an outer approximation for a large class of binary quadratic programs. *Operations Research*, 2022.
- [50] Martin Bergner, Alberto Caprara, Alberto Ceselli, Fabio Furini, Marco E Lübbecke, Enrico Malaguti, and Emiliano Traversi. Automatic Dantzig-Wolfe reformulation of mixed integer programs. *Mathematical Programming*, 149(1-2):391–424, 2015.
- [51] Taghi Khaniyev, Samir Elhedhli, and Fatih Safa Erenay. Structure detection in mixed-integer programs. *INFORMS Journal on Computing*, 30(3):570–587, 2018.
- [52] Enrico Bettiol, Immanuel Bomze, Lucas Létocart, Francesco Rinaldi, and Emiliano Traversi. Mining for diamonds—matrix generation algorithms for binary quadratically constrained quadratic problems. *Computers & Operations Research*, 142:105735, 2022.

- [53] Hao Hu and Renata Sotirov. The linearization problem of a binary quadratic problem and its applications. *Annals of Operations Research*, 307(1):229–249, 2021.
- [54] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.
- [55] Duy Tan Nguyen, Yossiri Adulyasak, Jean-Francois Cordeau, and Silvia I Ponce. Data-driven operations and supply chain management: established research clusters from 2000 to early 2020. *International journal of production research*, 60(17):5407–5431, 2022.
- [56] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, 2015.
- [57] James A Anderson. *An introduction to neural networks*. MIT press, 1995.
- [58] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.
- [59] Dina Bentayeb, Nadia Lahrichi, and Louis-Martin Rousseau. Patient scheduling based on a service-time prediction model: a data-driven study for a radiotherapy center. *Health care management science*, 22(4):768–782, 2019.
- [60] Peyman Kafeai, Quentin Cappart, Marc-Andre Renaud, Nicolas Chapados, and Louis-Martin Rousseau. Graph neural networks and deep reinforcement learning for simultaneous beam orientation and trajectory optimization of cyberknife. *Physics in Medicine & Biology*, 66(21):215002, 2021.
- [61] Umberto Junior Mele, Luca Maria Gambardella, and Roberto Montemanni. Machine learning approaches for the traveling salesman problem: A survey. In *2021 The 8th International Conference on Industrial Engineering and Applications (Europe)*, pages 182–186, 2021.
- [62] Chao Ning and Fengqi You. Data-driven stochastic robust optimization: General computational framework and algorithm leveraging machine learning for optimization under uncertainty in the big data era. *Computers & Chemical Engineering*, 111:115–133, 2018.
- [63] Arnaud Augustin, Philippe Jouvét, Nadia Lahrichi, Andrea Lodi, and Louis-Martin Rousseau. A data-driven approach to include availability of icu beds in the planning of the operating room. *Omega*, 109:102608, 2022.

- [64] Dimitris Bertsimas and Aurélie Thiele. Robust and data-driven optimization: modern decision making under uncertainty. In *Models, methods, and applications for innovative decision making*, pages 95–122. INFORMS, 2006.
- [65] Dick den Hertog and Krzysztof Postek. Bridging the gap between predictive and prescriptive analytics-new optimization methodology needed. *Tilburg Univ, Tilburg, The Netherlands*, 2016.
- [66] Dimitris Bertsimas and Nathan Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.
- [67] Kris Johnson Ferreira, Bin Hong Alex Lee, and David Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & service operations management*, 18(1):69–88, 2016.
- [68] Toon Vanderschueren, Tim Verdonck, Bart Baesens, and Wouter Verbeke. Predict-then-optimize or predict-and-optimize? an empirical evaluation of cost-sensitive learning strategies. *Information Sciences*, 594:400–415, 2022.
- [69] Adam N Elmachtoub and Paul Grigas. Smart “predict, then optimize”. *Management Science*, 68(1):9–26, 2022.
- [70] Sheng Liu, Long He, and Zuo-Jun Max Shen. On-time last-mile delivery: Order assignment with travel-time predictors. *Management Science*, 67(7):4095–4119, 2021.
- [71] David Bergman, Teng Huang, Philip Brooks, Andrea Lodi, and Arvind U Raghunathan. Janos: an integrated predictive and prescriptive modeling framework. *INFORMS Journal on Computing*, 34(2):807–816, 2022.
- [72] Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343–367, 2017.
- [73] Nicolas Beldiceanu and Helmut Simonis. Modelseeker: Extracting global constraint models from positive examples. In *Data Mining and Constraint Programming*, pages 77–95. Springer, 2016.
- [74] Mohit Kumar, Samuel Kolb, Luc De Raedt, and Stefano Teso. Learning mixed-integer linear programs from contextual examples. *arXiv preprint arXiv:2107.07136*, 2021.
- [75] Christian Bessiere, Frédéric Koriche, Nadjib Lazaar, and Barry O’Sullivan. Constraint acquisition. *Artificial Intelligence*, 244:315–342, 2017.

- [76] Dimosthenis C Tsouros and Kostas Stergiou. Efficient multiple constraint acquisition. *Constraints*, 25(3):180–225, 2020.
- [77] Tomasz P Pawlak and Krzysztof Krawiec. Automatic synthesis of constraints from examples using mixed integer linear programming. *European Journal of Operational Research*, 261(3):1141–1157, 2017.
- [78] Elias Arnold Schede, Samuel Kolb, and Stefano Teso. Learning linear programs from data. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1019–1026. IEEE, 2019.
- [79] Adejuyigbe Fajemisin, Donato Maragno, and Dick den Hertog. Optimization with constraint learning: A framework and survey. *arXiv preprint arXiv:2110.02121*, 2021.
- [80] Didier Burton and Ph L Toint. On an instance of the inverse shortest paths problem. *Mathematical programming*, 53:45–61, 1992.
- [81] Timothy CY Chan, Rafid Mahmood, and Ian Yihang Zhu. Inverse optimization: Theory and applications. *arXiv preprint arXiv:2109.03920*, 2021.
- [82] Farzin Ahmadi, Fardin Ganjkanloo, and Kimia Ghobadi. Inverse learning: A data-driven framework to infer optimizations models. *arXiv preprint arXiv:2011.03038*, 2020.
- [83] Timothy CY Chan and Neal Kaw. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 282(2):415–427, 2020.
- [84] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.
- [85] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, 2014.
- [86] Maciej Drwal. Algorithm for quadratic semi-assignment problem with partition size coefficients. *Optimization Letters*, 8(3):1183–1190, 2014.
- [87] Han Shen, Lichao Huang, Chang Huang, and Wei Xu. Tracklet Association Tracker: An End-to-End Learning-based Association Approach for Multi-Object Tracking. 8 2018. URL <https://arxiv.org/abs/1808.01562>.

- [88] Patrick Emami, Panos M. Pardalos, Lily Elefteriadou, and Sanjay Ranka. Machine Learning Methods for Solving Assignment Problems in Multi-Target Tracking. 2 2018. URL <https://arxiv.org/abs/1802.06897>.
- [89] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.394.
- [90] Afshin Dehghan and Mubarak Shah. Binary quadratic programing for online tracking of hundreds of people in extremely crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):568–581, 2017.
- [91] Shayan Modiri Assari, Haroon Idrees, and Mubarak Shah. Human re-identification in crowd videos using personal, social and environmental constraints. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9906 LNCS:119–136, 2016. ISSN 16113349. doi: 10.1007/978-3-319-46475-6{_}8.
- [92] Laura Leal-Taixe, Gerard Pons-Moll, and Bodo Rosenhahn. Branch-and-price global optimization for multi-view multi-target tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1987–1994. IEEE, 2012.
- [93] Shaofei Wang, Steffen Wolf, Charless C. Fowlkes, and Julian Yarkony. Tracking objects with higher order interactions via delayed column generation. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2017.
- [94] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.
- [95] Marco E Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.
- [96] Hatem Ben Amor, Jacques Desrosiers, and Antonio Frangioni. *Stabilization in column generation*. Groupe d’études et de recherche en analyse des décisions, 2004.
- [97] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.
- [98] Olivier Du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1):229–237, 1999.

- [99] Walid Ben-Ameur and José Neto. Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks: An International Journal*, 49(1):3–17, 2007.
- [100] Louis-Martin Rousseau, Michel Gendreau, and Dominique Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.
- [101] Allyson Silva, Leandro C Coelho, and Maryam Darvish. Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search. *European Journal of Operational Research*, 292(3):1066–1084, 2021.
- [102] Michael Booth, Steven P. Reinhardt, and Aidan Roy. Partitioning optimization problems for hybrid classical/quantum execution. *GitHub repository*, 2017. URL https://github.com/dwavesystems/qbsolv/blob/master/qbsolv_techReport.pdf.
- [103] Alain Billionnet and Sourour Elloumi. Best reduction of the quadratic semi-assignment problem. *Discrete Applied Mathematics*, 109(3):197–213, 2001.
- [104] Ingmar Schüle, Hendrik Ewe, and Karl-Heinz Küfer. Finding tight RLT formulations for quadratic semi-assignment problems. In *Proceedings of 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, CTW*, pages 109–112, 2009.
- [105] Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. 3 2016. URL <https://arxiv.org/abs/1603.00831>.
- [106] Morton E O’Kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.
- [107] J. Fabian Meier, Uwe Clausen, Borzou Rostami, and Christoph Buchheim. A compact linearisation of euclidean single allocation hub location problems. *Electronic Notes in Discrete Mathematics*, 52:37 – 44, 2016. ISSN 1571-0653. doi: <http://dx.doi.org/10.1016/j.endm.2016.03.006>. URL <http://www.sciencedirect.com/science/article/pii/S1571065316300117>. INOC 2015 – 7th International Network Optimization Conference.
- [108] Harold S. Stone. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE transactions on Software Engineering*, (1):85–93, 1977.
- [109] Philippe Chrétienne. A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints. *European Journal of Operational Research*, 43(2):225–230, 1989.

- [110] Pierre Hansen and K-W Lih. Improved algorithms for partitioning problems in parallel, pipelined, and distributed computing. *IEEE Transactions on Computers*, 41(6):769–771, 1992.
- [111] Hiroo Saito, Tetsuya Fujie, Tomomi Matsui, and Shiro Matuura. A study of the quadratic semi-assignment polytope. *Discrete Optimization*, 6(1):37 – 50, 2009. ISSN 1572-5286. doi: <http://dx.doi.org/10.1016/j.disopt.2008.08.003>. URL <http://www.sciencedirect.com/science/article/pii/S1572528608000625>.
- [112] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565, 1976.
- [113] Fabio Furini, Emiliano Traversi, Pietro Belotti, Antonio Frangioni, Ambros Gleixner, Nick Gould, Leo Liberti, Andrea Lodi, Ruth Misener, Hans Mittelmann, et al. QPLIB: a library of quadratic programming instances. *Mathematical Programming Computation*, 11(2):237–265, 2019.
- [114] Lea Bottmer, Christophe Croux, and Ines Wilms. Sparse regression for large data sets with outliers. *European Journal of Operational Research*, 297(2):782–794, 2022.
- [115] Jack William Dunn. *Optimal trees for prediction and prescription*. PhD thesis, Massachusetts Institute of Technology, 2018.
- [116] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106:1039–1082, 2017.
- [117] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.
- [118] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
- [119] Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2012. URL <http://www.gurobi.com>.
- [120] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [121] Patrick De Causmaecker and Greet Vanden Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14:3–16, 2011.

- [122] Chong Man Ngoo, Say Leng Goh, Nasser R Sabar, Salwani Abdullah, Graham Kendall, et al. A survey of the nurse rostering solution methodologies: The state-of-the-art and emerging trends. *IEEE Access*, 2022.
- [123] Sara Ceschia, Nguyen Dang, Patrick De Causmaecker, Stefaan Haspeslagh, and Andrea Schaerf. The second international nurse rostering competition. *Annals of Operations Research*, 274(1-2):171–186, 2019.
- [124] Petter Strandmark, Yi Qu, and Timothy Curtois. First-order linear programming in a column generation-based heuristic approach to the nurse rostering problem. *Computers & Operations Research*, 120:104945, 2020.
- [125] Say Leng Goh, Nasser R Sabar, Salwani Abdullah, Graham Kendall, et al. A 2-stage approach for the nurse rostering problem. *IEEE Access*, 10:69591–69604, 2022.
- [126] Stefaan Haspeslagh, Patrick De Causmaecker, Martin Stølevik, and Andrea Schaerf. First international nurse rostering competition 2010 (august 10-13, 2010, belfast, uk). In *PATAT 2010-Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, Belfast, Northern-Ireland, UK*, 2010.
- [127] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

APPENDIX A SUPPLEMENTS OF CHAPTER 3

A.1 Linearization

The Standard Linearization Technique is a known technique in the literature to transform the BQP model into its equivalent Mixed Integer Programming by substituting the quadratic terms with extra binary variables y_{ef} . For instance, linearized constraints for the MOT problem are as follows:

$$y_{ef} \geq x_e + x_f - 1 \quad \forall (e, f) \in \mathcal{A} \quad (\text{A.1})$$

$$y_{ef} \leq x_e \quad \forall (e, f) \in \mathcal{A} \quad (\text{A.2})$$

$$y_{ef} \leq x_f \quad \forall (e, f) \in \mathcal{A} \quad (\text{A.3})$$

$$y_{ef} \geq 0 \quad \forall (e, f) \in \mathcal{A}, \quad (\text{A.4})$$

where it suggests $O(n^3)$ decision variables and constraints be added to the BQP model.

In addition, Reformulation Linearization Technique is another mathematical programming technique used for linearization. In this study, we exploit a tighter linearization which is similar to RLT to linearize the BQP formulation of MOT as well as linearizing the star-based reformulation of MOT. Considering (3.22) and multiplying it once by $x_f, \forall f \in \delta^{t'}(j)$, where t' represents any frame except frame t and then generating the same constraint for x_e , we obtain the following strong valid inequalities instead of SLT constraints (A.2) and (A.3):

$$\sum_{e \in \delta^t(j)} y_{ef} \leq x_f \quad \forall t \in T, \forall j \in H, \forall f \in \delta^{t'}(j) : t' \neq t \quad (\text{A.5})$$

$$\sum_{f \in \delta^t(j)} y_{ef} \leq x_e \quad \forall t \in T, \forall j \in H, \forall e \in \delta^{t'}(j) : t' \neq t \quad (\text{A.6})$$

The RLT model applied to the AQSAP of (3.10)-(3.12) is formulated as following:

$$\min \quad \sum_{e \in A} c_e x_e + \sum_{\substack{i, k \in N \\ i \neq k}} \sum_{\substack{e \in \delta(i) \\ f \in \delta(k)}} q_{ef} y_{ef} \quad (\text{A.7})$$

$$\text{s.t.} \quad \sum_{e \in \delta(i)} x_e = 1 \quad \forall i \in N \quad (\text{A.8})$$

$$\sum_{e \in \delta(i)} y_{ef} = x_f \quad \forall i, k \in N, i \neq k, \forall f \in \delta(k) \quad (\text{A.9})$$

$$\sum_{f \in \delta(k)} y_{ef} = x_e \quad \forall i, k \in N, i \neq k, \forall e \in \delta(i) \quad (\text{A.10})$$

$$y_{ef} \geq 0 \quad \forall i, k \in N, i \neq k, \forall e \in \delta(i), \forall f \in \delta(k) \quad (\text{A.11})$$

$$x_e \in \{0, 1\} \quad \forall e \in A. \quad (\text{A.12})$$

A.2 Instance-by-instance tables

In the following, we detail the computational results for each problem in two tables consisting of dual bounds and computational time. For each problem, one table demonstrates the experiments for the instances in which at least one of the compared methods stops within the time limit. The other table contains instances in which all the methods reach the time limit; thus, the three hours of fixed computation time is not reported in this table to avoid repetition. As an index of efficiency for each method, we report the ratio of the obtained lower bound to the Best-known Feasible Solution (BFS). The BFS is computed by comparing the obtained Upper Bounds of all methods in the three-hour timeframe.

Table A.1 AQSAP – Comparing different methods – At least one of the methods stops within the time limit

Instance (m-n)	BFS	BQP		CG+BQPPricing		CG+HeuristicBQPPricing		CG+HeuristicSLTPricing		SLT		CG+SLTPricing		RLT		OuterApproximation	
		LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time
10-3	3038.8	1	0	1	2	1	8	1	6	1	1	1	2	1	1	1	0.1
10-4	2829.3	1	0	1	4	1	11	1	8	1	2	1	1	1	2	1	0.1
15-3	32202.4	1	52	1	56	1	18	1	14	1	40	1	16	1	30	1	6
15-4	21336.7	1	524	1	57	1	28	1	28	1	314	1	17	1	593	1	120
15-6	4472.7	1	327	1	41	1	25	1	16	1	233	1	10	1	1395	1	95
18-3	8168.7	1	43	1	298	1	35	1	24	1	25	1	142	1	61	1	13
18-4	6747.3	1	1852	1	142	1	5	1	8	1	909	1	127	1	1910	1	535
18-6	5906.2	0.82	10800	1	96	1	53	1	43	0.81	10800	1	147	0.73	10800	0.82	10800
18-8	4916	0.85	10800	1	96	1	83	1	82	0.8	10800	1	160	0.74	10800	0.79	10800
20-3	15549.8	1	1099	1	430	1	10	1	25	1	345	1	531	1	422	1	68
20-4	13657.3	0.92	10800	1	333	1	58	1	25	0.94	10800	1	331	0.86	10800	1	7406
20-6	9781.5	0.72	10800	1	227	1	101	1	52	0.68	10800	1	293	0.55	10800	0.66	10800
20-8	8559.1	0.62	10800	1	105	1	10	1	31	0.62	10800	1	176	0.55	10800	0.62	10800
20-10	4810.8	0.83	10800	1	151	1	18	1	59	0.81	10800	1	195	0.61	10800	0.79	10800
22-3	42873.5	0.97	10800	1	1744	1	51	1	74	1	7360	1	1612	0.98	10800	1.00	4807
22-4	24085.5	0.87	10800	1	500	1	27	1	76	0.84	10800	1	590	0.75	10800	0.80	10800
22-6	18316.3	0.72	10800	1	237	1	28	1	66	0.63	10800	1	404	0.46	10800	0.52	10800
22-8	7394.2	0.74	10800	1	205	1	13	1	31	0.67	10800	1	293	0.61	10800	0.67	10800
22-10	6017.8	0.78	10800	1	188	1	95	1	49	0.73	10800	1	284	0.59	10800	0.73	10800
22-12	4519.2	0.83	10800	1	158	1	100	1	73	0.76	10800	1	197	0.56	10800	0.80	10800
25-3	42794.9	0.94	10800	1	3127	1	51	1	72	0.98	10800	1	2417	0.89	10800	0.92	10800
25-4	14012.7	0.9	10800	1	598	1	16	1	79	0.9	10800	1	656	0.82	10800	0.88	10800
25-6	9478.1	0.68	10800	1	1937	1	684	1	174	0.59	10800	1	1477	0.46	10800	0.54	10800
25-8	8807.1	0.68	10800	1	581	1	566	1	252	0.64	10800	1	614	0.47	10800	0.57	10800
25-10	6318.8	0.7	10800	1	975	1	98	1	99	0.61	10800	1	591	0.43	10800	0.56	10800
25-12	4744	0.61	10800	1	265	1	228	1	179	0.52	10800	1	348	0.38	10800	0.47	10800
30-3	48847.1	0.89	10800	1	1741	1	70	1	98	0.94	10800	1	1863	0.90	10800	1.00	8424
30-4	20369.8	0.74	10800	0.99	10800	1	3665	1	1143	0.73	10800	1	9636	0.62	10800	0.65	10800
30-6	14167	0.52	10800	1	10800	1	2001	1	2287	0.5	10800	1	10800	0.39	10800	0.45	10800
30-8	10322.5	0.58	10800	1	6378	1	2612	1	964	0.5	10800	1	2953	0.31	10800	0.45	10800
30-14	6680.8	0.47	10800	1	1875	1	214	1	831	0.41	10800	1	818	0.31	10800	0.38	10800
40-14	10365.5	0.4	10800	0.98	10800	1	10800	1	4062	0.34	10800	1	10800	0.21	10800	0.31	10800
50-4	49468.9	0.43	10800	1	10800	1	6895	1	10800	0.37	10800	0.97	10800	0.29	10800	0.35	10800
50-6	40831	0.67	10800	1	10800	1	4156	0.56	10800	0.25	10800	0.82	10800	0.17	10800	0.22	10800
50-8	34960.5	0.5	10800	1	10800	1	7146	0.46	10800	0.24	10800	0.78	10800	0.18	10800	0.22	10800

Table A.2 AQSAP – Comparing different methods – None of the methods stop within the time limit

Instance (m-n)	BFS	BQP	CG+BQPPricing	CG+Heuristic BQPPricing	CG+Heuristic SLTPricing	SLT	CG+SLTPricing	RLT	Outer Approximation
		LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS
40-3	46075.8	0.66	0.94	1	1	0.58	0.93	0.51	0.58
40-4	37264	0.46	0.89	0.80	0.80	0.41	0.89	0.28	0.35
40-6	17322.9	0.36	0.86	0.86	0.86	0.28	0.88	0.20	0.26
40-8	16018.9	0.34	0.94	0.84	1	0.29	0.90	0.21	0.27
50-3	52448.4	0.61	0.99	1	1	0.50	0.94	0.47	0.50
50-14	11696.4	0.29	0.99	1	0.41	0.34	0.78	0.21	0.29

According to the tables associated with the MOT problem, increasing the parameter d for a problem with a fixed number of frames ($\mathcal{T}$) results in more computational time, highlighting the effects of the density of the quadratic matrix in solving the problem. Table A.4 demonstrates that in many cases the CG algorithms converge to an optimal solution of the problem. Specifically, when we use the CG+RLTPricing and the problem can be solved within the time limit, optimality is proved for in almost all the instances.

A.3 Primal Bounding for Adjacent-only Problems

As mentioned in Chapter 3, primal bounding methodologies are beyond the scope of this project. However, to show the strength of the proposed reformulation framework and CG for the adjacent-only class of BQPs, we obtain the primal bounds for all of the methods. In the this section we discuss the obtained UB for the adjacent-only problems.

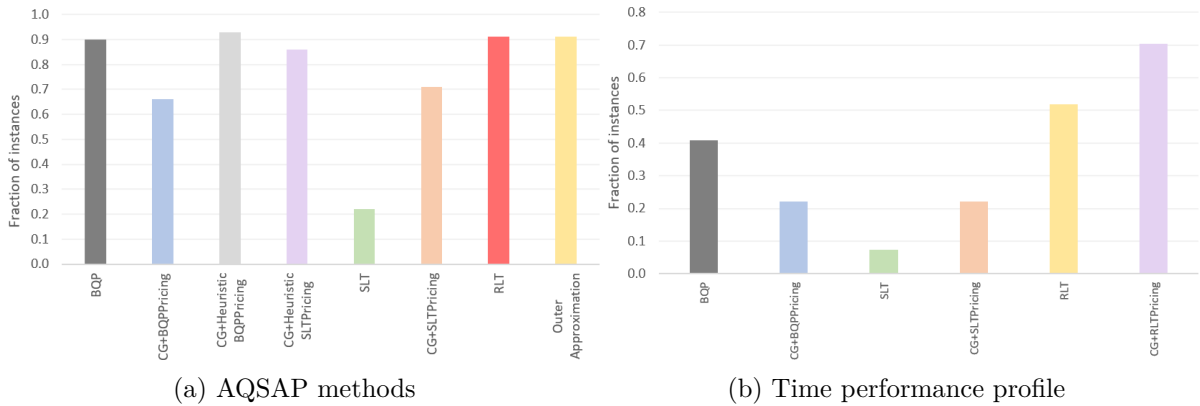


Figure A.1 Comparing methods in terms of obtaining BFS

Table A.3 Data association on MOT16-09 data set – Comparing GUROBI and CG – At least one of the methods stops within the time limit

Instance ($\mathcal{T}$ -h-d)	BFS	BQP			CG+BQPPricing			SLT			CG+SLTPricing			RLT			CG+RLTPricing		
		LB	LB/BFS	Time	LB	LB/BFS	Time	LB	LB/BFS	Time	LB	LB/BFS	Time	LB	LB/BFS	Time	LB	LB/BFS	Time
3-20-2	-266.4	-401.7	1.51	10800	-266.8	1	75	-266.4	1	1089	-266.8	1	64	-266.4	1	509	-266.8	1	12
4-25-2	-486.2	-1064.7	2.19	10800	-486.2	1	72	-486.2	1	6969	-486.2	1	950	-486.2	1	367	-486.2	1	70
4-25-3	-516.7	-1372.8	2.66	10800	-516.7	1	507	-551.9	1.07	10800	-538.2	1.04	10800	-516.7	1	5498	-516.7	1	78
5-25-2	-636.1	-1693.5	2.66	10800	-636.1	1	200	-665.3	1.05	10800	-636.1	1	4674	-636.1	1	5939	-636.1	1	466
5-25-3	-740.1	-1774.6	2.4	10800	-740.1	1	6522	-1477.4	2	10800	-740.1	1	6347	-754.8	1.02	10800	-740.1	1	555
5-25-4	-783.2	-2024.3	2.58	10800	-783.2	1	7291	-2177.8	2.78	10800	-783.2	1	5676	-806.7	1.03	10800	-783.2	1	400
6-25-3	-910.1	-2359	2.59	10800	-1365.8	1.5	10800	-2754.3	3.03	10800	-945.1	1.04	10800	-937.6	1.03	10800	-913.1	1	733
6-25-4	-1014.5	-2817	2.78	10800	-1898.7	1.87	10800	-3883	3.83	10800	-1070.2	1.05	10800	-1064.4	1.05	10800	-1014.5	1	822
6-25-5	-1110.1	-3087.7	2.78	10800	-2260.7	2.04	10800	-3493	3.15	10800	-2288.8	2.06	10800	-1137.2	1.02	10800	-1110.6	1	821
7-30-3	-1040.5	-2932.4	2.82	10800	-2509.7	2.41	10800	-3915.9	3.76	10800	-1135.1	1.09	10800	-1077.3	1.04	10800	-1040.8	1	1304
7-30-4	-1172.7	-3538.3	3.02	10800	-3038	2.59	10800	-4499	3.84	10800	-1267	1.08	10800	-1236.6	1.05	10800	-1172.8	1	1374
7-30-5	-1336.9	-3753	2.81	10800	-3496.2	2.62	10800	-5062.4	3.79	10800	-1456.3	1.09	10800	-1417.3	1.06	10800	-1336.9	1	1718
8-30-3	-1229.1	-3379.1	2.75	10800	-2261.9	1.84	10800	-4567.4	3.72	10800	-1741.7	1.42	10800	-1303.5	1.06	10800	-1229.7	1	4293
8-30-4	-1497.8	-4778.2	3.19	10800	-3806.2	2.54	10800	-5646.2	3.77	10800	-2002	1.34	10800	-1638.2	1.09	10800	-1497.8	1	4252
8-30-5	-1619.4	-5207.5	3.22	10800	-4373.9	2.7	10800	-6429.1	3.97	10800	-2902	1.79	10800	-1892.7	1.17	10800	-1632.4	1.01	5088

Table A.4 Data association on MOT16-09 data set – Comparing GUROBI and CG – None of the methods stops within the time limit

Instance (T-h-d)	BFS	BQP		CG+BQPPricing		SLT		CG+SLTPricing		RLT		CG+RLTPricing	
		LB	LB/BFS	LB	LB/BFS	LB	LB/BFS	LB	LB/BFS	LB	LB/BFS	LB	LB/BFS
9-35-4	-1790.8	-5601.5	3.13	-5377.9	3	-6569.6	3.67	-4230.2	2.36	-2026.3	1.13	-1792.1	1.00
9-35-5	-1974.7	-6624.1	3.35	NA	NA	-7740.1	3.92	-5085.4	2.58	-2358.4	1.19	-1988.9	1.01
9-35-6	-2065.4	-7647.6	3.7	NA	NA	-8519.1	4.12	-5134.7	2.49	-2535.4	1.23	-2150.4	1.04
10-35-4	-1971.7	-6472	3.28	NA	NA	-7357.6	3.73	-5469.1	2.77	-2288.2	1.16	-2073.7	1.05
10-35-5	-2150.6	-8116.3	3.77	NA	NA	-8663.3	4.03	-5700.6	2.65	-2665.1	1.24	-2346.1	1.09
10-35-6	-2405.1	-9273.5	3.86	NA	NA	-9792.8	4.07	-6824.3	2.84	-2988.5	1.24	-2621.2	1.09
10-35-7	-2569.7	-10134.5	3.94	NA	NA	-10681.3	4.16	-7544.9	2.94	-3252.3	1.27	-2833.3	1.10
11-35-4	-2167.5	-7494.8	3.46	-6969.1	3.22	-8306.2	3.83	-5996.4	2.77	-2650.4	1.22	-2412.8	1.11
11-35-6	-2687	-10773.8	4.01	-8849.4	3.29	-11098.4	4.13	-8436	3.14	-3440.9	1.28	-3074.5	1.14
11-35-9	-3141.3	-12752.7	4.06	-9512.7	3.03	-13306.3	4.24	-9496.8	3.02	-4107.9	1.31	-3540.1	1.13
12-40-4	-2467.1	-8257	3.35	-8606.6	3.49	-9107.1	3.69	-8168.3	3.31	-2915.7	1.18	-2862.4	1.16
12-40-6	-3044.7	-11709.1	3.85	-10676.2	3.51	-12016.3	3.95	-10281.8	3.38	-3762.6	1.24	-3520.2	1.16

Although we use a very trivial heuristic to find the feasible solutions after the CG methods terminate, in a relatively large fraction of instances in adjacent-only problems, the BFS is obtained through CG, meaning that the reformulation and CG outperform other formulations and methods in terms of both primal bound and dual bound for these problems.

In Figure A.1, we demonstrate a comparison of different methods for attaining the BFS for each instance. This figure proves the capabilities of the CG methods to find the best UBs in both the AQSAP and the MOT problem. According to Figure A.1a, the CG+HeuristicBQPPricing obtains the BFS for largest fraction of instances (93%) among all of the methods. For MOT, Figure A.1b indicates that the CG+RLTPricing is the best method, finding the BFS for 70% of instances.

APPENDIX B SUPPLEMENTS OF CHAPTER 4

B.1 Instance-by-instance tables

Similar to the instance-by instance tables in Appendix A, in this section, we present the details related to the QSAP experiments in terms of dual bounds and computational time. For each QSAP set of experiments, there is a table related to the instances in which at least one of the compared methods terminates within the predetermined time limit. There exists a second table demonstrating instances where all methods reach the time limit, therefore, we remove the column related to the fixed three-hour computation time to avoid redundancy. To assess the efficiency of each method, we provide a metric by calculating the ratio between the obtained lower bound and the Best-known Feasible Solution. The BFS is determined by comparing the UB obtained by all the methods within the three-hour timeframe.

A closer look at Tables B.1 to B.8 proves that in some cases of the QSAP with out-of-star interactions, one has to choose the solution methodology by considering the trade-off between time and the quality of LB. The reason is that the CG may stop in a shorter time with a slightly worse LB, which we do not consider in our performance analysis. In addition, for larger-size instances of different data sets in the QSAP, a significant pattern is observed in the results. When the number of servers (h) is relatively large compared to the number of clients (n), one of the CG methods yields the best results, while instances with a smaller number of servers are solved by GUROBI and the OuterApproximation methods with better bounds. Interestingly, in certain cases where none of the techniques stops prior to the time limit, CG exhibits a superior LB. As various techniques (such as acceleration and stabilization techniques) can be applied to further improve the performance of CG, there is room for enhancements in our methodology to make it more effective in tackling large-scale problems.

Table B.1 QSAP – 10% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit

Instance (m-n)	BFS	BQP		CG+BQPPricing		CG+ HeuristicBQPPricing		CG+ HeuristicSLTPricing		SLT		CG+SLTPricing		RLT		OuterApproximation	
		LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time
10-3	3061.4	1	0.1	1	6	1	6	1	3	1	2	1	2	1	0.1	1	0.1
10-4	2829.3	1	1	1	5	1	5	1	5	1	1	1	2	1	0.1	1	0.1
15-3	33075.4	1	4	0.98	77	0.98	10	0.98	6	1	11	0.98	22	1	2	1	1
15-4	21665.4	1	21	0.99	71	0.99	18	0.99	14	1	348	0.99	17	1	10	1	5
15-6	4497.2	1	7	0.99	40	0.99	15	0.99	15	1	79	0.99	64	1	5	1	6
18-3	8383	1	4	0.99	164	0.99	41	0.99	32	1	13	0.99	180	1	1	1	2
18-4	6884.3	1	23	0.98	144	0.98	29	0.98	24	1	119	0.98	169	1	12	1	25
18-6	5957.5	1	248	0.99	123	0.99	47	0.99	39	1	4021	0.99	135	1	239	1	395
18-8	4956.1	1	4762	0.99	125	0.99	43	0.99	40	0.76	10800	0.99	129	1	3748	1	6587
20-3	16245.3	1	20	0.97	533	0.97	103	0.97	55	1	75	0.97	561	1	25	1	5
20-4	13942.2	1	492	0.98	459	0.98	53	0.98	25	1	1701	0.98	502	1	170	1	37
20-6	9919.2	1	4867	0.99	224	0.99	42	0.99	23	0.51	10800	0.99	270	1	9593	1	2166
20-8	8697.8	0.75	10800	0.98	156	0.98	76	0.98	31	0.53	10800	0.98	250	0.71	10800	0.76	10800
20-10	4878.3	0.88	10800	0.99	194	0.99	87	0.99	34	0.64	10800	0.99	280	0.74	10800	0.81	10800
22-3	44517.4	1	444	0.96	1092	0.96	34	0.96	21	1	1632	0.96	1432	1	328	1	183
22-4	25071.6	1	4806	0.96	589	0.96	89	0.96	36	1	9336	0.96	768	1	3453	1	2033
22-6	18872.5	0.79	10800	0.97	312	0.97	89	0.97	31	0.55	10800	0.97	391	0.76	10800	0.70	10800
22-8	7587.9	0.82	10800	0.97	322	0.97	109	0.97	59	0.62	10800	0.97	511	0.75	10800	0.80	10800
22-10	6121.4	0.84	10800	0.99	336	0.99	84	0.99	107	0.65	10800	0.99	432	0.72	10800	0.78	10800
22-12	4566.7	0.88	10800	0.99	353	0.99	111	0.99	126	0.62	10800	0.99	478	0.71	10800	0.90	10800
25-3	44875.1	1	206	0.96	3207	0.96	516	0.96	176	1	918	0.96	4427	1	181	1	196
25-4	14811.7	1	83	0.95	1256	0.95	122	0.95	66	1	567	0.95	1298	1	184	1	153
25-6	9887.9	0.79	10800	0.96	2361	0.96	376	0.96	195	0.47	10800	0.96	2629	0.74	10800	0.79	10800
25-8	8999.1	0.73	10800	0.98	2051	0.98	673	0.98	402	0.53	10800	0.98	1492	0.64	10800	0.67	10800
25-10	6509.8	0.74	10800	0.97	2698	0.97	367	0.97	543	0.39	10800	0.97	1822	0.55	10800	0.63	10800
25-12	4921.6	0.47	10800	0.96	1130	0.96	1024	0.96	916	0.35	10800	0.96	1712	0.45	10800	0.57	10800
30-3	53502.1	1	203	0.93	8255	0.94	412	0.94	79	1	2902	0.94	1863	1	361	1	208
30-4	21896.3	1	1359	0.93	10800	0.93	4229	0.93	4350	1	8366	0.93	9636	1	1101	1	2049
30-6	14736.8	0.73	10800	0.95	10800	0.96	2324	0.96	3722	0.43	10800	0.96	10800	0.65	10800	0.64	10800
30-8	10807.5	0.40	10800	0.95	8560	0.95	4120	0.95	2720	0.24	10800	0.95	8120	0.44	10800	0.47	10800
30-14	6978.9	0.36	10800	0.96	7894	0.96	1063	0.96	10800	0.26	10800	0.96	10800	0.33	10800	0.39	10800

B.2 Upper Bound for QSAP

In this section we point out some discussions on how we provide a trivial primal bound for the QSAP. We apply a very simple heuristic to find the feasible solutions after the CG methods terminate. Akin to the heuristic introduced in Appendix A for AQSAP and MOT, here we use all of the entered columns in the QSAP-RMP to build our IP model to obtain a valid UB for QSAP. However, instead of solving the IP version of the last RMP directly, we solve an RMP with a slightly different objective function. In this model, we build a new quadratic cost function to reflect the interaction between each pair of stars (columns) of the RMP as below:

$$\begin{cases} Q_{s,s'} = 0, & \text{if } s \text{ and } s' \text{ have } i \text{ or } j \text{ in common} \\ Q_{s,s'} = \sum_{e \in s, f \in s'} q_{ef}, & \text{otherwise} \end{cases}$$

Therefore, the alternative model to solve is:

$$\min \sum_{s \in S} C_s \lambda_s + \sum_{s \in S} \sum_{s' > s \in S} Q_{ss'} \lambda_s \lambda_{s'} \quad \text{s.t.} \quad (3.15 - 3.16), \quad (\text{B.1})$$

where λ_s is a binary variable. through the new definition of quadratic costs, we discard many impossible interactions among stars from the beginning. This leads to reaching the same feasible solution as with the previously mentioned heuristic in Appendix A for adjacent BQPs, but in a much shorter time. Nonetheless, even with this reformulation the obtained feasible solutions are not promising enough.

Table B.2 QSAP – 10% out-of-star quadratic matrix density – Comparing different methods – None of the methods stops within the time limit

Instance (m-n)	BFS	BQP LB/BFS	CG+BQPPricing LB/BFS	CG+Heuristic BQPPricing LB/BFS	CG+Heuristic SLTPricing LB/BFS	SLT LB/BFS	CG+SLTPricing LB/BFS	RLT LB/BFS	Outer Approximation LB/BFS
40-3	51693.8	0.75	0.83	0.68	0.87	0.63	0.83	0.68	0.72
40-4	41684.2	0.52	0.78	0.71	0.71	0.22	0.78	0.40	0.48
40-6	19765.3	0.31	0.76	0.18	0.18	0.17	0.68	0.24	0.27
40-8	18865.9	0.21	0.43	0.35	0.35	0.17	0.43	0.23	0.26
40-14	11586.1	0.25	0.51	0.51	0.38	0.21	0.51	0.25	0.28
50-3	60996.3	0.83	0.87	0.88	0.82	0.58	0.79	0.73	0.65
50-4	62141.8	0.46	0.79	0.81	0.76	0.26	0.73	0.44	0.53
50-6	51281.2	0.27	0.78	0.79	0.27	0.16	0.50	0.18	0.23
50-8	44208.2	0.18	0.77	0.79	0.33	0.15	0.45	0.15	0.19
50-14	14426.3	0.22	0.37	0.25	0.30	0.20	0.29	0.17	0.23

Table B.3 QSAP – 15% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit

Instance (m-n)	BFS	BQP		CG+BQPPricing		CG+HeuristicBQPPricing		CG+HeuristicSLTPricing		SLT		CG+SLTPricing		RLT		OuterApproximation	
		LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time
10-3	3172.5	1	0	0.97	4	0.97	4	0.97	4	1	2	0.97	1	1	0	1	0
10-4	2881.7	1	1	0.99	5	0.99	5	0.99	4	1	1	0.99	1	1	0	1	0
15-3	33324.7	1	2	0.97	54	0.97	14	0.97	10800	1	8	0.97	18	1	1	1	1
15-4	22318.5	1	6	0.96	63	0.96	26	0.96	20	1	65	0.96	65	1	5	1	4
15-6	4568.5	1	4	0.98	41	0.98	16	0.98	21	1	38	0.98	56	1	3	1	4
18-3	8669.3	1	2	0.96	182	0.96	24	0.96	22	1	6	0.96	212	1	0	1	1
18-4	6978.7	1	10	0.97	194	0.97	29	0.97	26	1	38	0.97	193	1	4	1	13
18-6	6126.4	1	130	0.96	122	0.96	62	0.96	59	1	1544	0.96	137	1	75	1	136
18-8	5034.8	1	419	0.98	97	0.98	57	0.98	52	1	7584	0.98	122	1	500	1	840
20-3	16434.4	1	9	0.96	600	0.96	71	0.96	45	1	27	0.96	550	1	6	1	2
20-4	14529.7	1	97	0.94	349	0.94	32	0.94	19	1	788	0.94	444	1	76	1	19
20-6	10257.6	1	1707	0.95	206	0.95	57	0.95	25	0.85	10800	0.95	283	1	1724	1	391
20-8	9062.6	1	9703	0.95	227	0.95	84	0.95	36	0.6	10800	0.95	289	0.85	10800	0.85	10800
20-10	4965.2	1	4169	0.97	196	0.97	110	0.97	50	0.65	10800	0.97	315	1	10800	1	6248
22-3	45955.1	1	222	0.93	897	0.93	52	0.93	37	1	1037	0.93	1525	1	151	1	88
22-4	26366.2	1	899	0.91	464	0.91	121	0.91	44	1	4682	0.91	654	1	868	1	899
22-6	19512	1	8792	0.94	256	0.94	73	0.94	29	0.61	10800	0.94	433	0.89	10800	1	7364
22-8	8214.5	1	4946	0.9	525	0.90	106	0.90	78	0.64	10800	0.9	492	1	6424	1	8687
22-10	6349.8	0.85	10800	0.95	259	0.95	168	0.95	237	0.65	10800	0.95	368	0.85	10800	1	9966
22-12	4821.5	0.70	10800	0.93	433	0.93	160	0.93	131	0.65	10800	0.93	490	1	10800	1	8706
25-3	48713.8	1	117	0.89	4418	0.89	287	0.89	202	1	377	0.89	4186	1	117	1	129
25-4	16521.4	1	46	0.86	1685	0.86	121	0.86	75	1	241	0.86	1631	1	81	1	122
25-6	10827.9	1	7729	0.88	1932	0.88	384	0.88	191	0.51	10800	0.88	1895	1	7087	1	8188
25-8	9745.2	0.61	10800	0.9	4584	0.9	952	0.9	511	0.65	10800	0.9	2642	0.72	10800	0.76	10800
25-10	7063.3	0.59	10800	0.89	3336	0.89	942	0.89	840	0.39	10800	0.89	2130	0.61	10800	0.68	10800
25-12	4738.4	0.44	10800	0.85	1328	0.85	10156	0.85	10283	0.32	10800	0.85	10800	0.45	10800	0.58	10800
30-3	57082.6	1	93	0.88	10800	0.88	1751	0.88	264	1	1025	0.88	10800	1	103	1	118
30-4	24213	1	109	0.84	10800	0.85	2085	0.85	1132	1	2207	0.85	10800	1	398	1	322
30-6	18614.7	0.68	10800	0.77	10800	0.77	809	0.77	2739	0.41	10800	0.77	10800	0.69	10800	0.64	10800
30-8	14294	0.38	10800	0.78	5906	0.78	5366	0.78	3479	0.25	10800	0.78	8978	0.39	10800	0.44	10800
30-14	10284	0.31	10800	0.80	9593	0.80	1371	0.80	1365	0.23	10800	0.80	10800	0.29	10800	0.34	10800
40-14	17160.5	0.17	10800	0.36	10800	0.22	1540	0.19	10800	0.15	10800	0.37	10800	0.17	10800	0.2	10800

We do not report the details on UB of QSAP, but we would like to provide some remarks to direct future research on this topic:

Remark 6. GUROBI methods, and in particular BQP, attain the BFS in the vast majority of cases in QSAP data sets. However, as the tables of Appendix A suggest, it could not find the best LB and close the optimality gap for many instances of the problems, while CG methods can prove optimality in many instances of adjacent-only problems.

Remark 7. Interestingly, we observe that in the cases where CG methods terminate within the time limit, the obtained UB is fairly close to the BFS provided by GUROBI. In contrast, when the CG methods reach the three-hour time limit, they suggest a weak UB, while their associated LB is still better than GUROBI in a relevant subset of instances.

Table B.4 QSAP – 15% out-of-star quadratic matrix density – Comparing different methods – None of the methods stops within the time limit

Instance (m-n)	BFS	BQP	CG+BQPPricing	CG+Heuristic BQPPricing	CG+Heuristic SLTPricing	SLT	CG+SLTPricing	RLT	Outer Approximation
		LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS
40-3	55376	0.85	0.75	0.62	0.82	0.65	0.76	0.78	0.76
40-4	47234.8	0.68	0.70	0.64	0.64	0.32	0.69	0.41	0.50
40-6	23518.1	0.29	0.62	0.22	0.22	0.15	0.60	0.25	0.27
40-8	27886.8	0.16	0.29	0.17	0.17	0.13	0.29	0.17	0.20
50-3	65385.1	0.91	0.81	0.82	0.77	0.66	0.76	0.83	0.80
50-4	70572.9	0.57	0.69	0.70	0.66	0.29	0.63	0.54	0.58
50-6	62755.8	0.24	0.63	0.65	0.23	0.14	0.51	0.17	0.25
50-8	53024.2	0.16	0.63	0.66	0.32	0.13	0.35	0.14	0.18
50-14	26288.9	0.12	0.17	0.03	0.16	0.12	0.16	0.10	0.14

Table B.5 QSAP – 20% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit

Instance (m-n)	BFS	BQP		CG+BQPPricing		CG+ HeuristicBQPPricing		CG+ HeuristicSLTPricing		SLT		CG+SLTPricing		RLT		OuterApproximation	
		LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time
10-3	3213.4	1	0	0.96	5	0.96	3	0.96	3	1	1	0.96	1	1	0	1	0
10-4	3001.4	1	1	0.96	9	0.96	5	0.96	4	1	1	0.96	2	1	0	1	0
15-3	34726.1	1	2	0.93	60	0.93	11	0.93	10	1	6	0.93	21	1	1	1	1
15-4	23852.4	1	7	0.90	72	0.90	19	0.90	19	1	54	0.90	75	1	4	1	3
15-6	5132.6	1	5	0.87	52	0.87	21	0.87	21	1	33	0.87	66	1	2	1	4
18-3	9895.4	1	2	0.84	182	0.84	31	0.84	26	1	5	0.84	227	1	1	1	1
18-4	7357.1	1	6	0.92	174	0.92	44	0.92	35	1	19	0.92	174	1	2	1	5
18-6	6379.6	1	43	0.92	138	0.92	54	0.92	48	1	374	0.92	138	1	21	1	63
18-8	5446	1	228	0.90	116	0.90	55	0.90	48	1	3503	0.90	134	1	201	1	571
20-3	18140.5	1	8	0.87	533	0.87	189	0.87	13	1	39	0.87	627	1	4	1	1
20-4	15836	1	73	0.86	394	0.86	28	0.86	20	1	553	0.86	461	1	76	1	14
20-6	11201.2	1	1681	0.87	197	0.87	51	0.87	26	1	6065	0.87	286	1	1108	1	315
20-8	10102.4	1	2891	0.85	169	0.85	124	0.85	40	0.62	10800	0.85	219	1	6039	1	5047
20-10	5464.7	1	4997	0.88	242	0.88	167	0.88	71	0.69	10800	0.88	337	1	4995	1	5199
22-3	49327.2	1	248	0.87	1066	0.87	64	0.87	28	1	692	0.87	1693	1	153	1	81
22-4	28349	1	705	0.85	524	0.85	113	0.85	45	1	4661	0.85	659	1	832	1	510
22-6	21754.1	1	7202	0.84	214	0.84	115	0.84	46	0.61	10800	0.84	431	1	10800	1	6822
22-8	9714.9	0.71	10800	0.76	342	0.76	132	0.76	87	0.63	10800	0.76	470	1	4536	1	4872
22-10	7733.6	0.74	10800	0.78	322	0.78	243	0.78	263	0.61	10800	0.78	438	1	10800	1	7526
22-12	5884.6	0.66	10800	0.77	685	0.77	461	0.77	687	0.54	10800	0.77	818	0.78	10800	1	10800
25-3	51647	1	88	0.84	4549	0.84	218	0.84	149	1	321	0.84	4396	1	135	1	110
25-4	17712.6	1	21	0.80	1669	0.80	112	0.80	69	1	74	0.80	1812	1	51	1	71
25-6	12832.8	1	2871	0.74	2300	0.74	241	0.74	196	0.77	10800	0.74	2235	1	4243	1	6120
25-8	13192.7	0.55	10800	0.67	5744	0.67	542	0.67	384	0.41	10800	0.67	2463	0.68	10800	0.63	10800
25-10	8560.9	0.52	10800	0.74	2446	0.74	1284	0.74	957	0.31	10800	0.74	1967	0.57	10800	0.62	10800
25-12	7883.8	0.35	10800	0.60	1181	0.60	366	0.60	295	0.27	10800	0.60	1173	0.36	10800	0.44	10800
30-3	61853.4	1	48	0.81	10800	0.81	1105	0.81	237	1	257	0.81	10800	1	69	1	71
30-4	31188.4	1	222	0.65	10800	0.66	10800	0.66	1557	1	2317	0.66	10800	1	546	1	711
30-6	25956.1	0.52	10800	0.55	10800	0.55	2629	0.55	1965	0.34	10800	0.55	10800	0.59	10800	0.59	10800
30-8	19918.5	0.33	10800	0.52	10800	0.52	4722	0.52	2499	0.18	10800	0.52	7811	0.30	10800	0.38	10800
30-14	14062.6	0.19	10800	0.48	3044	0.48	1185	0.48	1191	0.15	10800	0.48	1461	0.17	10800	0.22	10800
50-3	71284.3	1	4427	0.75	10800	0.75	10800	0.69	10800	1	5015	0.37	10800	1	7429	0.86	10800

Table B.6 QSAP – 20% out-of-star quadratic matrix density – Comparing different methods
– None of the methods stops within the time limit

Instance (m-n)	BFS	BQP	CG+BQPPricing	CG+Heuristic BQPPricing	CG+Heuristic SLTPricing	SLT	CG+SLTPricing	RLT	Outer Approximation
		LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS
40-3	59514.1	0.85	0.71	0.65	0.76	0.74	0.71	0.82	0.80
40-4	53056.6	0.70	0.62	0.50	0.50	0.27	0.61	0.42	0.50
40-6	28760	0.27	0.53	0.29	0.29	0.14	0.50	0.24	0.26
40-8	39915	0.14	0.19	0.11	0.11	0.10	0.21	0.13	0.16
40-14	28785.3	0.10	0.18	0.17	0.10	0.10	0.19	0.10	0.12
50-4	77901.2	0.74	0.63	0.64	0.60	0.36	0.56	0.68	0.65
50-6	74931.2	0.29	0.53	0.55	0.22	0.14	0.33	0.18	0.23
50-8	70230.7	0.15	0.48	0.49	0.23	0.11	0.27	0.13	0.15
50-14	27759.7	0.12	0.17	0.01	0	0.12	0.15	0.11	0.14

Table B.7 QSAP – 25% out-of-star quadratic matrix density – Comparing different methods – At least one of the methods stops within the time limit

Instance (m-n)	BFS	BQP		CG+BQPPricing		CG+ HeuristicBQPPricing		CG+ HeuristicSLTPricing		SLT		CG+SLTPricing		RLT		OuterApproximation	
		LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time	LB/BFS	Time
10-3	3456.3	1	1	0.89	7	0.89	5	0.89	4	1	1	0.89	1	1	0	1	0
10-4	3161	1	1	0.91	6	0.91	8	0.91	6	1	0	0.91	2	1	0	1	0
15-3	37682.9	1	2	0.86	66	0.86	11	0.86	8	1	6	0.86	20	1	1	1	1
15-4	25935.9	1	6	0.82	77	0.82	26	0.82	22	1	53	0.82	89	1	3	1	3
15-6	6285.4	1	3	0.71	52	0.71	24	0.71	23	1	50	0.71	65	1	2	1	4
18-3	10703.9	1	2	0.77	168	0.77	37	0.77	38	1	5	0.77	221	1	0	1	1
18-4	9346.3	1	7	0.72	203	0.72	32	0.72	29	1	23	0.72	222	1	3	1	10
18-6	7765.1	1	76	0.76	124	0.76	51	0.76	41	1	722	0.76	142	1	27	1	100
18-8	6158.9	1	187	0.80	116	0.80	80	0.80	71	1	2245	0.80	131	1	156	1	551
20-3	20576.4	1	10	0.77	519	0.77	52	0.77	16	1	33	0.77	704	1	4	1	2
20-4	16772.2	1	34	0.82	462	0.82	51	0.82	22	1	154	0.82	712	1	21	1	10
20-6	12269.6	1	960	0.80	239	0.80	77	0.80	27	1	4011	0.80	285	1	725	1	165
20-8	11915.7	0.75	10800	0.72	175	0.72	174	0.72	43	0.61	10800	0.72	264	1	6101	1	2875
20-10	6446.6	1	10800	0.75	238	0.75	140	0.75	68	0.66	10800	0.75	346	1	3108	1	8047
22-3	50813.7	1	126	0.84	1037	0.84	42	0.84	42	1	307	0.84	1644	1	79	1	60
22-4	30416.8	1	621	0.79	376	0.79	95	0.79	54	1	1807	0.79	729	1	545	1	542
22-6	24468.6	1	9098	0.75	249	0.75	98	0.75	59	0.70	10800	0.75	360	1	6559	1	10800
22-8	11778.1	0.73	10800	0.63	336	0.63	125	0.63	90	0.66	10800	0.63	443	1	4523	1	5447
22-10	10706.9	0.54	10800	0.56	334	0.56	257	0.56	164	0.50	10800	0.56	491	0.67	10800	0.77	10800
22-12	7346.2	0.59	10800	0.61	590	0.61	812	0.61	723	0.46	10800	0.61	973	0.76	10800	1	10800
25-3	54414.9	1	59	0.80	3991	0.80	368	0.80	283	1	150	0.80	4858	1	102	1	74
25-4	19097.4	1	22	0.75	2513	0.75	175	0.75	75	1	45	0.75	2209	1	39	1	59
25-6	15827.4	0.79	10800	0.60	3070	0.60	378	0.60	131	0.78	10800	0.60	1609	1	5183	1	3281
25-8	16652.1	0.47	10800	0.53	3526	0.53	969	0.53	475	0.37	10800	0.53	1659	0.60	10800	0.58	10800
25-10	13344.2	0.40	10800	0.47	1363	0.47	389	0.47	271	0.22	10800	0.47	1273	0.42	10800	0.46	10800
25-12	12815	0.21	10800	0.37	8340	0.37	251	0.37	195	0.19	10800	0.37	1505	0.24	10800	0.31	10800
30-3	65491.3	1	28	0.77	10800	0.77	904	0.77	272	1	190	0.77	10800	1	18	1	27
30-4	35820.9	1	95	0.57	10800	0.57	6451	0.57	6451	1	1277	0.57	10800	1	212	1	349
30-6	33203.3	0.51	10800	0.43	10800	0.43	7491	0.43	3041	0.29	10800	0.43	10800	0.60	10800	0.54	10800
30-8	27684.8	0.27	10800	0.37	8469	0.37	1600	0.37	2229	0.15	10800	0.37	6692	0.29	10800	0.31	10800
30-14	18797.5	0.15	10800	0.36	9217	0.36	1482	0.36	1422	0.12	10800	0.36	3186	0.14	10800	0.19	10800
40-14	43918.3	0.07	10800	0.11	10800	0.1	1540	0.07	10800	0.06	10800	0.11	10800	0.07	10800	0.09	10800
50-3	74787.6	1	236	0.71	10800	0.72	10800	0.71	10800	1	2195	0.66	10800	1	2001	0.94	10800
50-4	78766.4	1	6021	0.62	10800	0.63	10800	0.59	10800	0.44	10800	0.55	10800	0.79	10800	0.83	10800

Table B.8 QSAP – 25% out-of-star quadratic matrix density – Comparing different methods
– None of the methods stops within the time limit

Instance (m-n)	BFS	BQP	CG+BQPPricing	CG+Heuristic BQPPricing	CG+Heuristic SLTPricing	SLT	CG+SLTPricing	RLT	Outer Approximation
		LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS	LB/BFS
40-3	65784.3	0.96	0.66	0.58	0.69	0.86	0.64	0.85	0.82
40-4	59743.9	0.72	0.56	0.42	0.42	0.34	0.55	0.45	0.50
40-6	32330.7	0.3	0.47	0.26	0.26	0.15	0.44	0.25	0.31
40-8	47935.6	0.13	0.19	0.15	0.33	0.09	0.17	0.11	0.16
50-6	77643.1	0.3	0.51	0.52	0.23	0.17	0.39	0.22	0.26
50-8	74816.2	0.14	0.46	0.44	0.24	0.11	0.25	0.11	0.16
50-14	27759.7	0.13	0.16	0.01	0	0.13	0.13	0.12	0.16

APPENDIX C SUPPLEMENTS OF CHAPTER 5

The MILP constraints which are added to the Nurse Rostering Problem as known rules are displayed in the following table.

C.1 Representation of tested Rules for Nurse Rostering Problem

Table C.1 MIP constraint representation of rules for Nurse Rostering Problem

Setting	Added MIP Constraints
Set1	$x_{i,3,d} + x_{i,1,d+1} \leq 1 \quad d : 1, \dots, p-1$
Set2	$2(1 - x_{i,3,d}) \geq x_{i,1,d+1} + x_{i,2,d+1} \quad d : 1, \dots, p-1$
Set3	$\begin{aligned} \sum_j (x_{i,j,7q-1} + x_{i,j,7q}) &\leq 5z_i + 0.001z_i + 0.999 && \forall i, \quad \forall q : 1, \dots, \lfloor p/7 \rfloor \\ \sum_j (x_{i,j,7q} + x_{i,j,7q+1}) &\leq 6(1 - z_i) && \forall i, \quad \forall q : 1, \dots, \lfloor p/7 \rfloor \\ \sum_j (x_{i,j,7q} + x_{i,j,7q+1}) &\geq 1 - z_i && \forall i, \quad \forall q : 1, \dots, \lfloor p/7 \rfloor \end{aligned}$