Hybrid Exact Methods for Solving Strictly Convex Integer Quadratic Programs

by

Wen-Yang Ku

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Abstract

Hybrid Exact Methods for Solving Strictly Convex
Integer Quadratic Programs

Wen-Yang Ku
Doctor of Philosophy
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2017

The strictly convex integer quadratically constrained problem (IQCP) is an important class of optimization problem that arises in numerous applications both in theoretical science and industry. Due to its NP-hard nature, common exact methods are typically tree search algorithms with various techniques for pruning sub-trees while ensuring that at least one optimal solution is retained. In this dissertation, we exploit the similarities among three tree search algorithms, discrete ellipsoid-based search (DEBS), mixed integer programming (MIP), and constraint programming (CP) and integrate their techniques to develop efficient hybrid algorithms for solving IQCPs. We empirically demonstrate the superior performance of our hybrid algorithms in comparison to the previous best known exact methods in the literature.

Our novel hybrid algorithms achieve the state-of-the-art on a variety of important classes of the IQCP, such as the binary quadratic programming problem, the exact quadratic knapsack problem, the variance minimization problem, and the integer least squares problems, which all have significant influence in the research fields of operations research (OR) and communications. In addition, we propose a novel global constraint in CP that infers domain reductions based on strictly convex quadratic constraints. Our new global constraint, together with the complementary branching heuristics, bring CP within an order of magnitude of the best-known techniques for the IQCPs tested, and achieve the state of the art when compared with general solvers such as CPLEX on some problem domains. Our new global constraint and branching heuristics fit naturally in the "model and solve" framework of CP and provide much more flexibility than the hand-crafted, specialized algorithms, allowing practitioners to easily adapt our techniques.

For the first time our study brings together the research of OR, CP and communications communities on IQCPs. We believe that our success in connecting different research fields can provide new insights into IQCPs and encourage future innovation.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor J. Christopher Beck, who has been continuously supportive throughout my PhD studies, especially during the difficult time in my life. Thank you for giving me the freedom to do research in the field of my interest, and for all the valuable opportunities to attend conferences and to work with industries.[1] Your deep knowledge and insights helped me at various stages of my research. I could not have finished the work without your patience and inspirational guidance.

I would also like to thank my committee members Professor Nikolaos V. Sahinidis, Professor Andre A. Cire, Professor Mark S. Fox, and Professor Dionne M. Aleman for their time, constructive comments and suggestions.

I would like to thank Tony, one of the brightest people I have met in my life, for his ideas, support and sharing the life as a PhD student. I am honored to have a friend like him.

Completing this work would have been much more difficult without the support and friendship from the TIDEL family. I would like to thank Chiara for always being encouraging, Michael for his wisdom, Margarita for always being available for discussions, Kyle for teaching me proper English, Chang for always being supportive, Eldan for his cool opinions about everything, Stefana for her cheerful spirit, and Ranjith for his kindness. I am grateful for all the helpful discussions, coffee breaks, and the surprise after my defense. In here I have met some of the best friends in my life, and I am so fortunate to be part of TIDEL.

I would like to thank all of the former TIDEL members for their comments and discussions. Especially, I would like to thank Maliheh for her advice on how to get through a PhD, and her support as a good friend.

I would like to thank my sister for being there for me always, and I would like to thank my mother and grandmother for bringing me up with the greatest love of all. I think of you constantly and I hope that you are proud of my accomplishments and the person I have become.

Finally, I would like to thank my wife Tzu-Jung for her love and support for everything. I will remember our countless discussions on the ELLIPSOID constraint and all the ellipsoidal fruits that we cut (and ate) for verifying the theories. This PhD would not have been possible without you.

[1] It is not possible to enumerate all the things that I am grateful of due to page limit!

Contents

1 Introduction **1**

1.1 Research Outline . . . 3

1.1.1 Integration of MIP & DEBS and CP & DEBS . . . 3

1.1.2 Constraint Integer Programming for Variance Minimization . . . 3

1.1.3 CP for General IQCPs . . . 4

1.2 Contributions . . . 4

1.2.1 State-of-the-Art Hybrid Algorithms . . . 4

1.2.2 A New Global Constraint for Strictly Convex IQCPs . . . 5

1.3 Organization of Dissertation . . . 5

2 Preliminaries **6**

2.1 Notation . . . 6

2.2 The Strictly Convex Integer Quadratically Constrained Problem . . . 7

2.2.1 Problem Definition . . . 7

2.2.2 A Motivating Example: Wind Farm Optimization . . . 7

2.2.3 Equivalence of IQCP and Integer Least Squares . . . 8

2.3 Exact Methods for Solving IQCPs . . . 8

2.3.1 Mixed Integer Programming . . . 9

2.3.2 Constraint Programming . . . 13

2.3.3 Constraint Integer Programming . . . 20

2.3.4 Discrete Ellipsoid-based Search . . . 23

2.4 Summary . . . 28

3 Combining DEBS with MIP and CP for Solving IQCPs **29**

3.1 Introduction . . . 29

3.1.1 Contributions . . . 30

3.1.2 Organization . . . 31

3.2 Combining DEBS and MIP . . . 31

3.2.1 Literature Review . . . 31

3.2.2 Problem Definition . . . 32

3.2.3 The Hybrid Algorithm . . . 33

3.2.4 Experimental Results: ILS Problems . . . 36

3.2.5 Experimental Results: BQP Problems . . . 43

3.3 Combining DEBS with CP . . . 48
3.3.1 Literature Review . . . 48
3.3.2 Problem Definition . . . 48
3.3.3 The Hybrid Algorithm . . . 49
3.3.4 Experimental Results: EQKP Problems . . . 51
3.4 Conclusion . . . 53

4 Constraint Integer Programming for Variance Minimization: An Application to Load Balancing Problems 55
4.1 Introduction . . . 55
4.1.1 Contributions . . . 57
4.1.2 Organization . . . 57
4.2 The Load Balancing Nurse-to-Patient Assignment Problem (NPAP) . . . 58
4.2.1 Background . . . 58
4.2.2 Mathematical Models . . . 58
4.3 The Balanced Academic Curriculum Problem (BACP) . . . 61
4.3.1 Background . . . 61
4.3.2 Mathematical Models . . . 61
4.4 Augmented Global Constraints . . . 63
4.4.1 The Quadratic Constraint . . . 63
4.4.2 The Global Cardinality Constraint . . . 64
4.4.3 The Spread Constraint . . . 65
4.5 Branching Heuristics . . . 67
4.6 Experimental Results: NPAP . . . 68
4.6.1 Setup . . . 68
4.6.2 Test sets . . . 68
4.6.3 Results . . . 68
4.6.4 Discussion . . . 70
4.7 Experimental Results: BACP . . . 73
4.7.1 Test sets . . . 73
4.7.2 Results and Discussion . . . 73
4.8 Conclusion . . . 74

5 Constraint Programming for Strictly Convex Integer Quadratically Constrained Problems 76
5.1 Introduction . . . 76
5.1.1 Contributions . . . 77
5.1.2 Organization . . . 77
5.2 Background . . . 78
5.2.1 The Strictly Convex Integer Quadratically Constrained Problem (IQCP) . . . 78
5.2.2 Discrete Ellipsoid-based Search (DEBS) . . . 78
5.3 The Ellipsoid Constraint . . . 79
5.4 Filtering Algorithms for the Ellipsoid Constraint . . . 79
5.4.1 A Direct Quadratically Constrained Programming (QCP) Formulation . . . 79

5.4.2 Axis-Aligned Tangent Box Filtering (BOX) . . . 80
5.4.3 Approximate Bounds Consistency (ABC) Filtering . . . 82
5.4.4 Relative Strength of the Three Filtering Algorithms . . . 85
5.4.5 Filtering Algorithm for the Axis-Aligned Ellipsoid (AAE) . . . 85
5.5 Branching Rules . . . 86
5.6 Experimental Results . . . 86
5.6.1 Problem Sets . . . 87
5.6.2 Results of Experiment 1 . . . 88
5.6.3 Results of Experiment 2 . . . 90
5.7 Conclusion . . . 92

6 Conclusion **93**
6.1 Summary . . . 93
6.2 Contributions . . . 94
6.3 Future Work . . . 95
6.3.1 CP as a Basis for MINLP . . . 95
6.3.2 New Mechanisms in CP . . . 97

Appendices **98**

A Additional Results of Chapter 3 **99**
A.1 Complete Results of the ILS Problems . . . 99
A.2 Noise Analysis for the BQPs . . . 103

Bibliography **104**

Chapter 1

Introduction

This dissertation develops exact solution approaches for the strictly convex integer quadratically constrained problem (IQCP) problem, an important class of optimization problem that arises in many applications, including algorithmic number theory [13], cryptography [119], global positioning systems [146], wireless communications [6], and scheduling [94]. The IQCP problem is known to be NP-hard [148], which means that no known algorithm is able to solve it in polynomial time. Given its theoretical challenge and practical value, it is of great interest to develop efficient algorithms that can find the optimal solution and prove its optimality. Specifically, the goal is to find the assignment of values to variables for the following problem:

$$\min_{\boldsymbol{x}\in\mathcal{C}} \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x},$$

$$\mathcal{C} = \left\{ \boldsymbol{x} \in \mathbb{Z}^n : \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{M}_k \boldsymbol{x} + \boldsymbol{c}_k^\top \boldsymbol{x} \leq b_k, \forall k = 1, \ldots, m,\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \right\},$$

where $\boldsymbol{l} \in \mathbb{Z}^n$, $\boldsymbol{u} \in \mathbb{Z}^n$, $\boldsymbol{H}$, $\boldsymbol{M}_k, \forall k$ are symmetric positive definite [66].

Although every NP-hard problem can be solved by an exhaustive search, it is practically impossible to enumerate and check all feasible solutions for large instances, sometimes even for instances of fairly small size [160]. In order to overcome this difficulty, the search space needs to be explored in an efficient and organized way. A systematic manner to explore the search space is to conduct a tree search: Let the original problem be the root node with all the variables unassigned, a variable is chosen and *branched on* by imposing additional constraints on the variable. Each additional constraint forms a branch that leads to a sub-problem (child node) with all the constraints of its parent node and the additional constraint, where all the sub-problems together partition the search space so that the search remains complete. Each sub-problem therefore has a more restricted (smaller) feasible region after branching, making it potentially easier to solve. The sub-problems are recursively divided into smaller sub-problems when necessary, thus forming a branching tree. The search terminates when it can be proved that all unexplored nodes lead either to infeasiblility in the search space or to solutions which are no better than the best solution found so far. Under some conditions, it can be inferred that certain branching decisions can never lead to an optimal solution. In these cases we can simply cut off the corresponding sub-trees, therefore speeding up the search. It is not surprising that common exact methods for solving IQCPs are tree search type algorithms with various techniques for pruning sub-trees that are guaranteed to not contain the optimal solution. In the following paragraphs we introduce several efficient tree

search strategies that have been proposed in the research fields of operations research (OR), constraint programming (CP), and communications for solving IQCPs.

In operations research, the common generic approach to solving IQCPs exactly are the so-called *branch-and-cut* based mixed integer programming (MIP) solvers [29]. By bounding the objective value at each node via solving a relaxed problem and strengthening the relaxed problem formulation with cutting planes to further restrict the feasible space, MIP has been proven effective for attacking combinatorial optimization problems [81]. In the realm of mixed integer *linear* programming (MILP), tremendous algorithmic progress has been achieved over the last decades [27, 81], allowing practitioners to use MIP as an out-of-the-box solver to solve a significant number of non-trivial MILP instances. However, the techniques for solving mixed integer *nonlinear* programing problems have not reached the maturity of MILP, leaving a great room for improvement.

Constraint programming (CP) [133] is another general tree search framework for solving combinatorial optimization problems that can be applied to IQCPs. At each node in the search tree, inference algorithms, often encapsulated in the so-called *global constraints* [153], are used to remove domain values that violate the constraints, therefore pruning sub-trees with value assignments that are not part of any solution. CP originated in artificial intelligence and became a recognized research area in the late 1970s [14], and has proven to be successful for modelling and solving complex discrete optimization problems over the last decade [14, 16]. However, the techniques for solving quadratically constrained problems have not received much attention. There are only a few dedicated global constraints that are concerned with quadratic constraints. For example, the Spread constraint [124] enforces a specific quadratic relationship amongst a set of variables, their mean, and their standard deviation. The general quadratic constraints [52, 96], while being able to reason both convex and nonconvex quadratic functions, do not exploit the strictly convex nature of the IQCP.

In communications, IQCPs are most commonly solved with discrete ellipsoid-based search (DEBS) based on enumeration of integer points within the hyper-ellipsoid defining the feasible space [6, 73]. DEBS can be understood from a CP perspective as a form of CP with a static variable ordering heuristic and a dynamic value ordering heuristic, where the geometry of the ellipsoid induces an interval domain for each variable, inducing an inference algorithm that is used to remove values that do not belong to the ellipsoid. In contrast to MIP and CP, DEBS is a specialized tree search that was originally developed to solve only three types of the integer least squares (ILS) problem. Therefore, although DEBS has been successfully applied to a variety of problems in communications applications [73, 108, 146], it cannot be applied to general IQCPs problem, e.g., problems with linear constraints,[1] due to its specialized nature.

Despite the difference in the underlying theories of these three approaches, rich information can be extracted from one method to enhance the others. In this dissertation, we aim at bringing together the techniques that have been often studied independently by these three research communities.

Thesis Statement

Integrating techniques from DEBS, MIP and CP results in efficient hybrid algorithms that perform better than using each method individually for solving IQCPs. Abundant geometric information can be extracted from DEBS and formulated as inference techniques that can be used in any tree search algorithm to prune search space.

[1] See Chapter 3 for an extension of DEBS to handle general linear constraints.

1.1 Research Outline

In this section we outline our research.

1.1.1 Integration of MIP & DEBS and CP & DEBS

In Chapter 3 we show how MIP, CP and DEBS can be hybridized to solve some special cases of the IQCP efficiently.

The hybridization of MIP and DEBS can be regarded as a component enhancement, since different aspects of DEBS are incorporated into MIP as a presolving technique, cutting planes, and a primal heuristic. As test sets, we study the binary quadratic programming (BQP) problem [131] and the three types of the ILS problems: unconstrained, box-constrained, and ellipsoid-constrained problems [6]. We perform experiments on benchmark instances for these four problems and compare the performance of two MIP solvers, DEBS, the new hybrid algorithm, and the best known available approaches in the literature. Our results demonstrate that the new hybrid algorithm is competitive to the best known approaches and achieves the new state of the art for many cases.

The hybridization of CP and DEBS is done in a different way and it can be seen as a node processing enhancement, i.e., the additional inference from DEBS is used to reduce the domains of the variables during a CP search. Specifically, we generalize DEBS to incorporate general linear constraints, broadening the class of problems that DEBS can solve. As noted previously, DEBS can be understood as a specialized CP search where the geometry of the ellipsoid induces an interval domain for each variable. We use this insight to integrate linear constraints into DEBS so that the domains of the variables are further reduced by the intervals implied by the linear constraints. Although CP has not been known for solving IQCPs efficiently, we show that a combination with DEBS greatly improves the performance of CP. For test sets, we use the exact quadratic knapsack problem (EQKP), an important class of optimization problem with a number of practical applications. We demonstrate that our hybrid algorithm achieves state-of-the-art for solving the EQKP and a variation.

1.1.2 Constraint Integer Programming for Variance Minimization

In Chapter 4, we study a tighter integration than the hybrid algorithms in Chapter 3, combining MIP, CP and DEBS together for solving the variance minimization problem, an important class of IQCP.

The integration is done in the paradigm of Constraint Integer Programming (CIP) [3] which integrates CP and MIP solving techniques. Specifically, at each node, we perform MIP routines such as LP relaxation solving and CP routines such as constraint propagation. In addition, we strengthen the problem formulation with cutting planes based on a CP perspective on problem structure and the geometrical reasoning from DEBS. We demonstrate that the CIP approach can be used to efficiently solve the variance minimization problem. For experiments, we focus our study on two load balancing problems: the load balancing nurse-to-patient assignment problem (NPAP) and the balanced academic curriculum problem (BACP), which appear frequently in scheduling applications. Results show that our CIP model with problem-specific branching rules outperforms both our new mixed integer quadratic programming (MIQP) model in CPLEX and the previous state-of-the-art CP model for the NPAP. For the BACP, both our new MIQP model and CIP model with problem-specific heuristic outperform the previous best known CP model.

1.1.3 CP for General IQCPs

In Chapter 5, we develop a CP approach that builds upon the previous chapters to solve a much broader class of problems with general strictly convex quadratic constraints, at the same time allowing the incorporation of any side constraints that fit into the CP framework.

Inspired by the geometric reasoning exploited in DEBS, we introduce two novel techniques for solving strictly convex IQCPs with CP. First, we propose the ellipsoid constraint, a novel global constraint that filters variable domains with respect to strictly convex quadratic functions. We derive a direct quadratically constrained programming (QCP) formulation that achieves bounds consistency (BC), and two light-weight filtering algorithms that do not guarantee BC. Though it is natural to consider integer domains in CP, our filtering algorithm can be applied to variables with real domains, broadening its application to, for example, mixed integer programming solvers. Second, we propose a pair of variable/value selection rules. We implement the filtering algorithms and the branching heuristics in IBM ILOG CP Optimizer. Our results on five problem domains show orders of magnitude improvement compared to the default CP Optimizer. When compared to the best known algorithms, our results demonstrate that the new CP approach is competitive to the best known approaches and establishes a new state of the art for some problem domains.

1.2 Contributions

1.2.1 State-of-the-Art Hybrid Algorithms

Strictly Convex Quadratically Constrained Problems with Variable Bounds. We propose a novel, competitive hybrid algorithm that combines DEBS and MIP and demonstrate that it is one of the state-of-the-art approaches for strictly convex quadratically constrained problems with variable bounds, i.e., problems without linear constraints.

We experiment with the binary quadratic programming (BQP) problem, an important class of problems in the operations research literature, and the ILS problems studied in the signal processing literature and demonstrate that our hybrid algorithm is the state-of-the-art approach. A particular advantage of this hybrid approach is that, unlike the specialized approaches (e.g., the semi-definite programming approach [87]) that only solve BQPs, it can be applied to a broader class of problems, such as the unconstrained IQCPs or IQCPs with general integer variable bounds. In addition, we conduct the first extensive experiments that compare the traditional OR approaches and DEBS on BQPs and the three types of ILS problems. We believe such a connection between MIP and DEBS can provide new insights into the IQCPs and may inspire future innovation. This work appeared in the CPAIOR2014 conference [89].

Strictly Convex Quadratically Constrained Problems with Linear Constraints. For the first time in the literature, we apply DEBS to problems with linear constraints. Our hybridization with CP enables DEBS to solve a much broader class of problems. As a test set, we experiment with the EQKP and a variation and achieve the state-of-the-art. This work is also the first study that analyzes the DEBS method from a CP perspective: We describe DEBS as a set of branching heuristics and a propagation algorithm. This work was originally published in the CPAIOR2015 conference [90].

Variance Minimization Problems. We show how various global constraints can be extended to their *augmented* forms to not only embed their traditional filtering algorithms but also linear relaxations strengthened with cutting planes. Within the framework of the augmented global constraint, we develop relaxations and cutting planes for the spread constraint and implement them in the CIP solver SCIP [142]. In terms of reusability, the augmented global constraints can be added to any solver that solves linear relaxations and supports the dynamic addition of cutting planes. As a test set, we apply our CIP approach for variance minimization with the application in load balancing problems. We also study the structure of the load balancing problems and propose problem-specific branching rules that guide the search in an efficient way. Our CIP models with branching rules outperform the previous best known non-decomposition based CP model for the nurse to patient assignment problem (NPAP) and the best known CP model for the balanced academic curriculum problem (BACP). In addition, we propose novel MIQP models that solve the load balancing problems in the CP literature and perform the first empirical study on the NPAP and the BACP that analyze the performance of MIP and CP. A preliminary report on this work has previous published in the CP2014 conference [93].

1.2.2 A New Global Constraint for Strictly Convex IQCPs

For the first time in the literature, we propose a CP approach for solving strictly convex IQCPs. We make a strong connection between CP and MINLP and bring a problem often seen as intractable to the CP community and incorporate it effectively in the framework of global constraints.

Our novel global constraint, the ellipsoid constraint, can be used to filter the variable domains for any formulation that contains a strictly convex quadratic function. We formally define the classical bounds consistency notation for the ellipsoid constraint and propose three inference algorithms with different strength of filtering power and computational complexity. In additional to the new global constraint, we develop variable and value ordering heuristics that complement our global constraint. Our novel CP approach brings CP within an order of magnitude of the state-of-the-art techniques for the IQCPs tested. When compared to non-problem specific solvers, our CP approach even achieves the state of the art on some problem types. We believe that this work can attract the OR community to the CP way of thinking and encourage further exploration of using CP as a basis for solving MINLPs. The use of our new global constraint and branching heuristics, as opposed to hand-crafted, specialized algorithms, provides more flexibility and fits well into the "model and solve" framework, allowing practitioners to utilize our techniques effortlessly. The work in this chapter is based on the CP2016 conference publication [91].

1.3 Organization of Dissertation

In Chapter 2 we describe the notation and necessary background for this dissertation, providing an overview of how the existing literature is related to our new approaches for solving IQCPs. In Chapter 3 we present our new hybrid algorithms that combine DEBS, MIP and CP for solving some special cases of the IQCP. In Chapter 4 we develop new CIP and MIQP approaches for variance minimization with an application to load balancing problems. In Chapter 5 we present our novel global constraint and branching heuristics for solving general IQCPs. Finally, we conclude and discuss possible future research in Chapter 6.

Chapter 2

Preliminaries

As described in the Introduction, in this dissertation we develop efficient hybrid algorithms for solving the strictly convex integer quadratically constrained problem (IQCP). In this chapter we describe the notation and necessary background for this dissertation. The relevant work is organized into two parts:

1. The problem: The strictly convex integer quadratically constrained problem (IQCP).

2. The methods: The exact solution approaches for solving IQCPs.

For the problem, we first give the formal definition of the IQCP and describe a real world example. We then make a connection between the IQCP and an equivalent formulation, the integer least squares (ILS) formulation. Such mutual reformulation is crucial for developing the hybrid algorithms in this dissertation. For the methods, we first describe the common exact methods for solving IQCPs and then explain the details of the methods that are relevant to our novel hybrid algorithms. Finally, we end this chapter with an overview of how the existing literature is related to our new approaches for solving IQCPs.

2.1 Notation

Although most notation is introduced in place, we summarize some common notation used in the dissertation for ease of reference.

The sets of all real and integer $m \times n$ matrices are denoted by $\mathbb{R}^{m\times n}$ and $\mathbb{Z}^{m\times n}$, respectively, and the set of real and integer n-vectors are denoted by $\mathbb{R}^n$ and $\mathbb{Z}^n$, respectively. Bold upper case letters denote matrices and bold lower case letters denote vectors. The identity matrix is denoted by $\boldsymbol{I}$ and its i-th column is denoted by $\boldsymbol{e}_i$. $\boldsymbol{A}_i$ denotes the i-th column of a matrix $\boldsymbol{A}$. $\boldsymbol{A}^\top$ and $\boldsymbol{A}^{-1}$ denote the transpose and inverse of a matrix $\boldsymbol{A}$, respectively. $\boldsymbol{A}_{ij}$ denotes the entry in the i-th row and j-th column of a matrix $\boldsymbol{A}$. $\|\cdot\|_2$ denotes the 2-norm of a vector or a matrix. $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_n)$ denotes a diagonal matrix. For a scalar $z \in \mathbb{R}$, we use $\lfloor z \rceil$ to denote its nearest integer. (If there is a tie, $\lfloor z \rceil$ denotes the one with smaller magnitude.) $\lfloor z \rfloor$ denotes its nearest integer less than or equal to z, and $\lceil z \rceil$ denotes its nearest integer greater than or equal to z.

2.2 The Strictly Convex Integer Quadratically Constrained Problem

The strictly convex integer quadratically constrained problem (IQCP) is an optimization problem where the objective and/or some constraints are strictly convex quadratic functions. The IQCP is an extension of integer linear programming [36] such that it allows for a more natural modelling of problems that can be represented with quadratic functions or a combination of linear and quadratic functions. As mentioned in Chapter 1, the IQCP has a wide range of applications such as algorithmic number theory [13], cryptography [119], global positioning systems [146], wireless communications [6], and scheduling [94].

2.2.1 Problem Definition

The general IQCP problem has the following form:

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{Z}^n} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x}, && \\ \text{s.t.} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{M}_k\boldsymbol{x} + \boldsymbol{c}_k^\top \boldsymbol{x} \leq b_k, && k = 1,\ldots,m, \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, && \end{aligned} \tag{2.1}$$

where $\boldsymbol{l} \in \mathbb{Z}^n$, $\boldsymbol{u} \in \mathbb{Z}^n$, $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{M}_k \in \mathbb{R}^{n\times n}$, $\boldsymbol{f} \in \mathbb{R}^n$, $\boldsymbol{c}_k \in \mathbb{R}^n$, and $b_k \in \mathbb{R}$. The IQCP is strictly convex if the quadratic matrices $\boldsymbol{H}$, $\boldsymbol{M}_k, \forall k$ are symmetric positive definite [66]. As the above form suggests, quadratic functions can appear in the objective function and/or the constraints.

2.2.2 A Motivating Example: Wind Farm Optimization

In this section we describe a real world scenario, the wind farm optimization (WFO) problem, which can be modelled as an IQCP. The WFO problem [147, 163] is concerned with placing wind turbines at different locations in the wind farm in order to maximize the energy generated. It is defined by a finite set of n locations where at most one turbine can be placed. For each location i, a non-negative value a_i is used to represent the cost of placing a turbine at location i. The total cost incurred from the turbines cannot exceed $MaxCost$. The objective can be approximated well with a quadratic function that describes the summation of power generated by the turbines considering the interrelation, a non-negative value h_{ij}, between each pair of the turbines, i.e., turbine at location i and j. The goal of the WFO problem is to place exactly K ($\leq n$) turbines on the wind farm in order to maximize the total energy generated.

Let the binary decision variable x_i be 1 if a turbine is placed at location i, the WFO problem can be modelled as an IQCP in Fig. 2.1.

The quadratic objective function is stated in (2.2). Constraint (2.3) specifies that exactly K turbines are chosen. Constraint (2.4) ensures that the total cost lies within the budget limit.

The WFO problem actually has the same structure as the *exact quadratic knapsack problem*, which is discussed in detail in Section 3.3.

$$\max_{\boldsymbol{x}\in\{0,1\}} \quad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} h_{ij}x_i x_j, \tag{2.2}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i = K, \tag{2.3}$$

$$\sum_{i=1}^{n} a_i x_i \leq MaxCost, \tag{2.4}$$

$$x_i \in \{0,1\}, \qquad i = 1,\ldots,n$$

Figure 2.1: The IQCP model for the WFO problem.

2.2.3 Equivalence of IQCP and Integer Least Squares

The integer least squares (ILS) problem can be defined as follows:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{2.5}$$

where $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ is a matrix with full column rank and $\boldsymbol{y}\in\mathbb{R}^m$ is a vector.

The strictly convex quadratic objective function (2.1) can be transformed from the equivalent ILS formulation (2.5) as follows:

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 &= \min_{\boldsymbol{x}\in\mathbb{Z}^n} (\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})^\top(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}),\\ &= \min_{\boldsymbol{x}\in\mathbb{Z}^n} \boldsymbol{x}^\top\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{y}^\top\boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^\top\boldsymbol{y},\\ &= \min_{\boldsymbol{x}\in\mathbb{Z}^n} \frac{1}{2}\boldsymbol{x}^\top\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top\boldsymbol{x},\end{aligned}$$

where $\boldsymbol{H}=\boldsymbol{A}^\top\boldsymbol{A}$ and $\boldsymbol{f}=-\boldsymbol{y}^\top\boldsymbol{A}$. Obviously, a quadratic constraint can be transformed in the same manner. It follows that the objective function of an ILS problem can be transformed from the equivalent IQCP problem through the relationship $\boldsymbol{A}=chol(\boldsymbol{H})$ and $\boldsymbol{y}=-(\boldsymbol{f}\boldsymbol{A}^{-1})^\top$, where *chol* denotes the Cholesky decomposition on a matrix [66].

Throughout this dissertation, we make use of this transformation to combine techniques from different research fields.

2.3 Exact Methods for Solving IQCPs

In this section we review the common exact methods for solving the strictly convex IQCP. Since IQCPs appear in numerous applications, they have been studied by many different research communities, often, independently. In this section we take a broad view of the solution approaches that are generally used in operations research (OR), constraint programming (CP), and communications.

In operations research, the common generic approaches to solving IQCPs exactly are the use of mixed integer nonlinear programming (MINLP) solvers such as BARON [134, 145], BONMIN [29] and ANTIGONE [112], and the application of MIP solvers such as CPLEX and Gurobi which have been extended to reason about convex quadratic constraints. We clearly describe how the techniques originally

developed for mixed integer linear programming (MILP) are generalized for IQCPs in Section 2.3.1. We end the MIP section with a discussion of the more general solving framework MINLP, and a specialized approach for binary problems, semi-definite programming (SDP), that is closely related to MIP.

In constraint programming, techniques to solve quadratically constrained problems have not receive much attention. There are only a few dedicated global constraints that reason about quadratic terms. For example, the Spread constraint [124] enforces the standard quadratic relationship amongst a set of variables, their mean, and their standard deviation. General quadratic constraints [52, 96] can be applied to both convex and nonconvex quadratic functions. However, they do not exploit the strictly convex nature of the IQCP. We briefly describe these global constraints in Section 2.3.2.3 and Section 2.3.2.4.

In communications, there are four main families of approaches to solving the ILS problems: the Voronoi approach [110], the approximation approach [7], semi-definite programming [85], and discrete ellipsoid-based search (DEBS) [82]. However, the Voronoi approach is known to be computationally inefficient, and the approximation approach and the semi-definite programming approach are not able to guarantee optimal solutions. Therefore in practice, ILS problems are most commonly solved with DEBS based on the enumeration of integer points within the hyper-ellipsoid defining the feasible space [6, 73]. We describe the details of DEBS in Section 2.3.4.

2.3.1 Mixed Integer Programming

The history of MILP dates back to the 1950s [81]. In recent years, substantial progress has been made and it has been shown that MILP techniques can be successfully applied to the convex mixed integer quadratically constrained problem (MIQCP), an important sub-class of the MINLP problem, where the quadratic objectives and/or constraints are convex [23, 31]. In this section we describe how a MIQCP is solved with MIP.

Formally, a *mixed integer program* is defined as follows [97].

Definition 2.3.1. A mixed integer program is an optimization problem of the form:

$$\min\{f(\boldsymbol{x}) \mid g_k(\boldsymbol{x}) \leq 0, \forall k = 1, \ldots, m,\ \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}],\ x_j \in \mathbb{Z}, \forall j \in I\},$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g_k : [\boldsymbol{l}, \boldsymbol{u}] \to \mathbb{R}$ are the constraint functions, and $I \subseteq N = \{1, \ldots, n\}$ is the index set of the integer variables.

Following Definition 2.3.1, a *mixed integer linear program* and a *mixed integer quadratically constrained program* are defined as follows:

- Mixed integer linear program: A mixed integer program is a mixed integer linear program if the objective function and the constraints are linear, i.e., $f(\boldsymbol{x})$ is of the form $\boldsymbol{c}^\top \boldsymbol{x}$, where $\boldsymbol{c} \in \mathbb{R}^n$, and $g_k(\boldsymbol{x}), \forall k = 1, \ldots, m$ are of the form $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, and $\boldsymbol{b} \in \mathbb{R}^m$.
- Mixed integer quadratically constrained program: A mixed integer program is a mixed integer quadratically constrained program if the objective function or one or more of the constraints are quadratic, i.e., $f(\boldsymbol{x})$ is of the form $\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x}$ and $g_k(\boldsymbol{x})$ is of the form $\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{M}_k\boldsymbol{x} + \boldsymbol{c}_k^\top \boldsymbol{x} - b_k$, where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$, $\boldsymbol{M}_k \in \mathbb{R}^{n \times n}$, $\boldsymbol{f} \in \mathbb{R}^n$, $\boldsymbol{c}_k \in \mathbb{R}^n$, and $b_k \in \mathbb{R}$. A mixed integer quadratically constrained program is convex if the quadratic matrices $\boldsymbol{H}$, $\boldsymbol{M}_k, \forall k = 1, \ldots, m$ are positive semi-definite [155].

We define the continuous relaxation of a mixed integer program as follows.

Definition 2.3.2. The continuous relaxation of a mixed integer program is defined as

$$\min\{f(\boldsymbol{x}) \mid g_k(\boldsymbol{x}) \leq 0, k = 1, \ldots, m,\ \boldsymbol{x} \in [\boldsymbol{l}, \boldsymbol{u}],\ \boldsymbol{x} \in \mathbb{R}^n\}.$$

Note that the integrality restrictions on x_j, $\forall j \in I$ are removed in the relaxed problem.

2.3.1.1 The Convex Mixed Integer Quadratically Constrained Problem

State-of-the-art MINLP solvers typically solve MIQCPs using three approaches: outer-approximations with MILP relaxations [54], branch and cut (B&C) with outer-approximation linear programming (LP) relaxation [127], and branch and bound with nonlinear programming (NLP) relaxation (in our case, quadratic programming (QP) relaxation) [101]. In this section we focus on the second approach, the outer-approximation based B&C approach, as it is the technique that many modern MIP solvers use. The basic idea of this approach is to exploit the already rich B&C MILP techniques, building the solver upon the B&C framework for the MILP, and then making the generalization to MIQCP problems [23]. A comprehensive survey on different approaches can be found in Bussieck and Vigerske [37].

We note that we are interested in the *strictly* convex IQCP, which is a sub-class of the convex MIQCP. Compared to the convex MIQCP where the quadratic matrices need only be positive semi-definite, the quadratic matrices in the strictly convex IQCP are required to be positive definite.

2.3.1.2 The Outer-approximation Based B&C Algorithm

Branch and Cut We briefly summarize the B&C method in this section. More details can be found in survey papers about the B&C method [1, 12, 107].

The basic B&C procedure is described as follows: Let P_0 be the original problem and the root node of the search tree. The first step of the algorithm is to consider a relaxation, $\bar{P}_0$, commonly the continuous relaxation of P_0 that disregards the integrality constraints on the integer variables. Let $\bar{\boldsymbol{x}}$ be an optimal solution for the relaxed problem $\bar{P}_0$. If $\bar{\boldsymbol{x}}$ is integer then we have found an optimal solution for the original problem and the algorithm terminates. Otherwise, a *branching heuristic* is used to choose one of the variables, x_i, whose value in the relaxed optimal solution, $\bar{x}_i$, is non-integer. This variable is "branched on" to create two sub-problems (children nodes): one with $x_i \geq \lceil \bar{x}_i \rceil$ and the other one with $x_i \leq \lfloor \bar{x}_i \rfloor$. It is clear that the feasible regions of the sub-problems are smaller after branching, thus making them potentially easier to solve. Another heuristic (for *node selection*) is used to choose which of the so far unexplored nodes should be processed next. As with the root node, the first step is to solve the relaxed problem. The sub-problems are further recursively divided into smaller sub-problems when necessary, thus forming a branching tree. The search terminates when the algorithm proves that all open nodes lead either to infeasiblility in the search space or to solutions which are no better than the best solution found so far (the *incumbent*).

In addition, at each node, after the relaxed problem is solved, hyperplanes of the form $\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{a}^\top \boldsymbol{x} = \alpha$ such that $\boldsymbol{a}^\top \bar{\boldsymbol{x}} > \alpha$ and $\boldsymbol{a}^\top \boldsymbol{x} \leq \alpha$ may be added to separate the optimal relaxed solution from the rest of the feasible region, cutting off part of the solution space without removing the optimal integer solution. This addition of *cutting planes* strengthens the relaxation and has become a key part of modern MILP solvers [27]. The B&C algorithm is so named because the incorporation of the cutting plane generation into the branching scheme.

Many other techniques crucial to the search efficiency can be added to the B&C process. For example, before the B&C process, presolving algorithms [3] can be executed to simplify the problem and extract information useful for the main search. In addition, during the search, primal heuristics algorithms [3] can be used to locate feasible solutions by heuristically constructing a candidate solution.

Outer Approximation The outer approximation [54] approach solves an MIQCP by iterating between solving a MILP problem and a continuous convex QP problem, until optimality conditions are reached. Let P denote the original MIQCP. The outer-approximated MILP problem P^{OA} is obtained from P by replacing the non-linear objective function and constraints with polyhedral outer-approximations. The convex QP is formed by fixing the integer variables of P to an optimal solution to P^{OA}. After solving the convex QP, additional information in the form of linear inequalities derived from the optimal solution to the QP problem is then used to strengthen P^{OA}. This approach has been shown to converge in a finite number of iterations if the original problem is convex and the objective function and constraints are differentiable [58].

Integration of Outer Approximation and Branch and Cut It has been shown that combining B&C and outer-approximation provides advantages over using either of them alone [127]. The B&C process is strengthened by using the outer-approximation method as an additional source of cutting planes. As before, the LP relaxation is solved at each node in the search tree. In addition, new linear outer-approximations of the original MIQCP are generated whenever the optimal solution of the LP relaxation is integer and these new linearizations are then lazily added as cutting planes during the search to improve the relaxations. Finally, integrality constraints are enforced by branching on the integer variables that take continuous values in the relaxation solution. The outer-approximation based B&C approach has all the advantages of the B&C process plus additional dynamic strengthening of the linear approximations. Compared to the pure outer-approximation method, the combination avoids solving the already difficult MILP problems multiple times.

We give a pseudo-code for the outer-approximation based B&C algorithm in Algorithm 1 due to [127]. Let P_0 be the original problem, i.e., the root node. We denote the problem (node) that is currently being solved as P_c. Let $\bar{P}_c$ be the continuous relaxation problem of P_c, $\bar{\boldsymbol{x}}$ be the optimal solution to $\bar{P}_c$, and $\text{obj}(\bar{\boldsymbol{x}})$ be the objective value that corresponds to $\bar{\boldsymbol{x}}$. We further let *incumbent* be the best candidate integer solution found so far and $\text{obj}(\textit{incumbent})$ be the objective value that corresponds to *incumbent*. We use $LIST$ as a data structure to store the tree nodes to be explored.

Recent advances in cutting planes techniques [10, 154] have significantly improved the performance of commercial MIQCP solvers such as CPLEX [31].

We note that modern MIP solvers can choose from two different node processing strategies, i.e., in addition to using the outer approximation based B&C algorithm that solves regular linear relaxation at each node, CPLEX can be set to solve quadratic relaxation directly for IQCPs. Empirical results show that neither of these two strategies dominates the other in general [47].

2.3.1.3 Mixed Integer Nonlinear Programming

Mixed integer nonlinear programming (MINLP) is concerned with problems with general nonlinear objective functions and constraints [97]. It is a further extension of the MILP problem and MIQCP. MINLP is an active research field that receives much attention from industry where many problems

Algorithm 1 Outer-approximation Based B&C Algorithm

Set $incumbent \leftarrow NULL$, obj($incumbent$) $\leftarrow \infty$
Add P_0 to $LIST$
while length($LIST$)> 0 **do**
  Choose P_c from $LIST$, solve $\bar{P}_c$ for $\bar{\boldsymbol{x}}$ and obj($\bar{\boldsymbol{x}}$)
  if $\bar{P}_c$ is feasible **then**
    if obj($\bar{\boldsymbol{x}}$) $\geq$ obj($incumbent$) **then**
      Cut off P_c, since there are no integer solutions to the problems in the subtree of P_c better than $incumbent$
    else if $\bar{\boldsymbol{x}} \in \mathbb{Z}^n$ **then**
      Compute linear outer-approximations and add them as cutting planes to the problems in $LIST$.
      obj($incumbent$) $\leftarrow$ obj($\bar{\boldsymbol{x}}$)
      $incumbent \leftarrow \bar{\boldsymbol{x}}$
    else
      Choose a variable x_i such that $\bar{x}_i \notin \mathbb{Z}$ to branch on
      Add the two sub-problems $P_c \cup \{x_i \geq \lceil \bar{x}_i \rceil \}$ and $P_c \cup \{x_i \leq \lfloor \bar{x}_i \rfloor \}$ to $LIST$
    end if
  else
    Cut off P_c, since it is infeasible
  end if
end while

cannot be approximated well with lower order functions [37, 63]. In addition to convex MINLP problems, a large body of the literature studies the nonconvex [35, 97] MINLP problem, which is much more difficult to solve than convex MINLPs, due to the typically nonconvex continuous NLP relaxation. It has been shown that solving a nonconvex NLP relaxation is in general NP-hard [117], which means that no known algorithm is able to locate the global optimum of the relaxation problem in polynomial time [144]. In order to overcome this difficulty, many techniques have been developed, such as piecewise linear approximations, convex relaxations for nonconvex functions, spatial branch and bound, and primal heuristics [35]. For a review on the techniques for convex MINLP and nonconvex MINLP problems, please refer to [30, 63, 69].

2.3.1.4 Semi-definite Programming-based Branch and Bound

Another generic approach for solving IQCPs is semi-definite programming (SDP) based branch and bound [87, 131]. SDP based approaches were originally developed to solve the max-cut problem [65]. The key idea is to replace the rather weak linear relaxation with the SDP relaxation, then the bound obtained from solving the SDP relaxation is used in the branch and bound tree search for pruning nodes. Several works have improved the strength of the SDP relaxation, which enables SDP solvers to achieve state-of-the-art performance on several classes of binary quadratic programming problems [88, 130] that can be reformulated from the max-cut formulation. Since the SDP relaxation is specifically developed for the max-cut problem, the available SDP solvers, e.g., BiqCrunch [88], only solve problems with binary variables as opposed to general integer variables.

2.3.2 Constraint Programming

Constraint Programming (CP) originated in artificial intelligence and became a recognized research area in the late 1970s [14]. It has proven to be successful for modelling and solving complex discrete optimization problems over the last decade [14, 16]. In the rest of the section, we introduce the necessary background for this dissertation based on the books *Constraint Processing* [50] and *Handbook of Constraint Programming* [133].

2.3.2.1 Constraint Satisfaction and Constraint Optimization

In CP, a combinatorial problem is often formulated as a *constraint satisfaction problem* (CSP) [133], which is represented by a set of variables, each with a set of possible values, and a set of constraints among the variables. The goal is to decide whether there exists a set of variable assignments that satisfies all the constraints. Formally, a CSP is defined as follows.

Definition 2.3.3. An instance of CSP can be formally described as a triple of $(X, D, \mathcal{C})$ where $X = \{x_1, x_2, ..., x_n\}$ each taking its value from a finite domain, $x_i \in D(x_i) \subset \mathbb{Z}, 1 \leq i \leq n$, and a finite set of constraints $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ each defined on a subset of the variables, $c_j(x_{j_1}, x_{j_2}, \ldots, x_{j_k}) \subset \mathbb{Z}^k, 1 \leq j \leq m$. A constraint $c_j(x_{j_1}, x_{j_2}, \ldots, x_{j_k})$ is defined on the Cartesian product of the domains of the variables in its scope $D(x_{j_1}) \times \ldots \times D(x_{j_k})$. A solution to an instance of CSP is an n-tuple assignment $A = (a_1, a_2, \ldots, a_n)$ where $a_i \in D(x_i)$ such that each c_j is satisfied.

It is often of interest to find the optimal solution w.r.t. some objective while satisfying all the constraints. A *constraint optimization problem* (COP) is a CSP with an objective function $f : D(x_1) \times \ldots \times D(x_n) \to \mathbb{R}$ that evaluates the assignment of values to the variables. An optimal solution to a COP is a solution to its corresponding CSP that minimizes, or maximizes, the value of f.

Typically, a COP is solved as a series of CSPs: When an incumbent is found during the search, a constraint that limits the next solution to be no worse than the current incumbent is added to the problem and search backtracks from that node. The search proves that the latest solution is optimal when no more solutions can be found.

2.3.2.2 Constraint Propagation

The key feature of CP is that the constraints are not only used to verify the feasibility of a solution but also to actively reduce the search space. This process is referred to as constraint propagation and is typically applied at each node of the search tree. Constraint propagation relies on inference, often in the form of propagation algorithms (or filtering algorithms). A filtering algorithm examines the variables that are not yet instantiated, and removes values that cannot participate in any solution w.r.t. the already instantiated variables. Specifically, given a constraint $c(x_1, x_2, \ldots, x_k)$, the propagation algorithm of c removes domain values from its variables when it can be proved that a value does belong to a solution of c, given the current domains of the other variables in the scope of c. The filtering algorithms are commonly executed for all constraints until a fixed point when no more domain reductions can be inferred.

Consistency. In order to formalize the concept of constraint propagation and characterize propagation algorithms, the notation of consistency is used to define the properties that a constraint problem must satisfy after constraint propagation. We describe the classical consistency notations [50] in this section.

Let x_j be a finite-domain variable, $D(x_j)$ be the domain of x_j, which is a set of values that can be assigned to x_j, and $I_D(x_j) = [l_j, u_j]$ be the interval domain of x_j. We first define arc consistency (AC) as follows.

Definition 2.3.4. A constraint $c(x_1, x_2)$ is arc consistent with respect to domains $D(x_1)$ and $D(x_2)$ if for every value $v_j \in D(x_1)$, there exists a value $v_i \in D(x_2)$ such that $c(x_1 = v_j, x_2 = v_i)$ holds, and for every value $v_j \in D(x_2)$, there exists a value $v_i \in D(x_1)$ such that $c(x_1 = v_j, x_2 = v_i)$ holds.

We note that AC is defined on a binary constraint. When a constraint is defined on a set of variables, AC is extended to hyper-arc consistency (HAC), which is also referred to as generalized arc consistency (GAC) or domain consistency (DC) [133]. We define DC as follows.

Definition 2.3.5. A constraint $c(x_1, \ldots, x_n)$ is domain consistent with respect to domains $D(x_j)$ if for all $j \in 1, \ldots, n$ and every value $v_j \in D(x_j)$, there exists values $v_i \in D(x_i)$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ such that $c(x_1 = v_1, \ldots, x_n = v_n)$ holds.

Achieving DC is often computationally expensive and NP-hard for some constraints [24]. Alternatively, bounds consistency (BC) is a weaker notion of consistency that can often be achieved more cheaply than DC for the same constraint. BC is defined as follows.

Definition 2.3.6. A constraint $c(x_1, \ldots, x_n)$ is bounds consistent with respect to domains $D(x_j)$ if for all $j \in 1, \ldots, n$ and each value $v_j \in \{l_j, u_j\}$, there exists values $v_i \in I_D(x_i)$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ such that $c(x_1 = v_1, \ldots, x_n = v_n)$ holds.

Global Constraints. Unlike the constraints in MIP that must take the form of a polynomial function, a constraint in CP can be specified intensionally with a formula that must be satisfied, or extensionally with a list of satisfying assignments of values to the variables in the constraint [133]. Although a constraint can be used to define any arbitrary relation among a set of variables, it is highly desirable to develop a constraint for recurring patterns that frequently arise in many problems, as such a constraint is reusable and can simplify modeling effort. In CP, such a constraint that captures an important combinatorial substructure of the problem is referred to as a *global constraint*. A global constraint is often semantically redundant in the sense that the same relation can be captured with a combination of simpler constraints, however, it often achieves more domain reductions [133]. For a comprehensive introduction to global constraints, please see van Hoeve and Katriel [153].

We explain the concept of the global constraint with the ALLDIFFERENT constraint, a well-known global constraint that specifies that the values assigned to the variables $x_1, \ldots, x_n$ must take different values. The formal definition of the ALLDIFFERENT constraint is given as follows.

Definition 2.3.7. Let $x_1, \ldots, x_n$ be a set of finite-domain variables, then

$$\text{ALLDIFFERENT}(x_1, \ldots, x_n) = \{(d_1, \ldots, d_n) \mid d_i \in D(x_i), \forall i, d_i \neq d_j, \forall i \neq j\}.$$

We now explain how to model the n-Queens problem with CP using the ALLDIFFERENT constraint [153]. The n-Queens problem can be defined with a $n \times n$ matrix representing the n rows and the n columns of a chess board, and n integer variables $x_1, \ldots, x_n$, where each $x_i \in \{1 \ldots, n\}$ represents the column location of the queen at the i-th row.

The goal of the n-Queens problem is to assign a number to each x_i such that no queen attacks another queen. Specifically, the no-attack constraints can be written in the pairwise not-equal constraints as follows:

$$x_i \neq x_j, \quad 1 \leq i < j \leq n, \tag{2.6}$$
$$x_i - x_j \neq i - j, \quad 1 \leq i < j \leq n, \tag{2.7}$$
$$x_i - x_j \neq j - i, \quad 1 \leq i < j \leq n, \tag{2.8}$$
$$x_i \in \{1, \ldots, n\}, \quad 1 \leq i \leq n.$$

Constraint (2.6) ensures that no two queens can be placed in the same column. Constraints (2.7) and (2.8) state that no two queens can be placed in the same diagonals. The same set of constraints can be modelled with the ALLDIFFERENT constraint as follows:

$$\begin{aligned} &\textsc{alldifferent}(x_1, \ldots, x_n), \\ &\textsc{alldifferent}(x_1 - 1, \ldots, x_n - n), \\ &\textsc{alldifferent}(x_1 + 1, \ldots, x_n + n), \\ &x_i \in \{1, \ldots, n\}, \quad 1 \leq i \leq n. \end{aligned}$$

Although the pairwise distinct property can easily be modeled with a set of not-equals between each pair of the variables, the filtering algorithm for the ALLDIFFERENT constraint can achieve more domain reduction [152] by providing a better view of the problem structure. Consider a 3-queen problem with initial domains $x_1 \in \{1, 2\}$, $x_2 \in \{1, 2\}$, $x_3 \in \{1, 2\}$. Achieving AC on (2.6) does not prune any of the domains from the variables. However, it is clear that there exists no solution for this problem. The filtering algorithm of the ALLDIFFERENT constraint [152] can identify such infeasibility and thus terminates the search immediately.

In the following two sections we describe the two global constraints that are concerned with quadratic constraints.

2.3.2.3 The Spread Constraint

The SPREAD constraint enforces a given mean μ and maximum standard deviation σ among a set of n variables $\{x_1, \ldots, x_n\}$. It can be defined as follows:

$$\texttt{spread}(\{x_1, \ldots, x_n\}, \mu, \sigma),$$

where $\sigma = \sqrt{\sum_{i=1}^{n}(x_i - \mu)^2/n}$ and $\mu = \sum_{i=1}^{n} x_i/n$.

Bounds consistency for the SPREAD constraint is defined separately for continuous and integer domains as follows [138]:

Definition 2.3.8. Let $I_D^{\mathbb{Q}}(x_i) = [l_i, u_i]$ be the rational interval domains of x_i, and let $I_D^{\mathbb{Z}}(x_i) = \{l_i, \ldots, u_i\}$ be the integer interval domains of x_i. A SPREAD constraint is $\mathbb{Q}$-bounds consistent (resp. $\mathbb{Z}$-bounds consistent) with respect to the domains of $\{x_1, \ldots, x_n\}$ if for all $i \in \{1, \ldots, n\}$ and each value $v_i \in \{l_i, u_i\}$, there exists values $v_j \in I_D^{\mathbb{Q}}(x_j)$ (resp. $v_j \in I_D^{\mathbb{Z}}(x_j)$) for all $j \in \{1, \ldots, n\} \setminus \{i\}$ such that $\texttt{spread}(\{x_1 = v_1, \ldots, x_n = v_n\}, \mu, \sigma)$ holds.

In other words, in $\mathbb{Z}$-BC, the support vector must be integer while in $\mathbb{Q}$-BC the support vector may be rational.

The SPREAD constraint was originally proposed by Pesant et al. [124], where a $\mathbb{Q}$-BC algorithm that filters the domains of the x_i variables w.r.t. constant μ and variable σ was presented. A simplified $\mathbb{Q}$-BC algorithm was presented in Schaus et al. [137, 138], where an extended algorithm that filters the mean variable was proposed. The $\mathbb{Z}$-BC algorithm was introduced in Schaus et al. [138], demonstrating stronger filtering than $\mathbb{Q}$-BC. Both of these existing BC algorithms address the case where μ is fixed, a natural restriction in the applications that have been studied, for example, where a fixed amount of work is to be divided as evenly as possible amongst a set of workers [139]. A $\mathbb{Q}$-BC filtering algorithm for the variable mean case was presented in our recent work [104]. For achieving DC, Pesant [121] proposed a pseudo-polynomial time filtering algorithm for the fixed mean case, where the generalization to the variable mean case was also discussed, showing that DC with variable mean can be achieved at a higher computational cost compared to the fixed mean case.

As the fixed mean $\mathbb{Q}$-BC filtering algorithm is used in Chapter 4 for solving the variance minimization problems, we describe the propagation algorithm due to Schaus et al. [137, 138] as follows.

Let $I(x)$ be the set of intervals defined by pairs of consecutive elements of the sorted sequence of bounds of the variables $\{x_1, \ldots, x_n\}$, we denote the k-th interval of $I(x)$ by I_k. Let $R(I_k) = \{x_i | l_i \geq \max(I_k)\}$ be the set of variables whose values must be greater than I_k. In a number line representation, $R(I_k)$ are the variables whose values will necessarily lie to the right of I_k. Similarly, $L(I_k) = \{x_i | u_i \leq \min(I_k)\}$ is the set of variables that lie to the left of I_k. We let $M(I_k)$ be the remaining variables not in $R(I_k)$ or $L(I_k)$ and let $m = |M(I_k)|$. We further let $ES(I_k)$ be the sum of the assigned x_i values at their extremes in $R(I_k)$ and $L(I_k)$, i.e., $ES(I_k) = \sum_{x_i \in R(I_k)} l_i + \sum_{x_i \in L(I_k)} u_i$.

Let $q = n\mu = \sum_{i=1}^{n} x_i$, we can compute $V(I_k) = [ES(I_k) + (m)\min(I_k), ES(I_k) + (m)\max(I_k)]$, which is the interval of possible q values related to I_k. Let $I^q \in I(x)$ such that $q \in V(I^q)$. That is, I^q is the interval that contains q. To find the minimum consistent value of σ, all the variables in $R(I^q)$ should be assigned to their minimum value, l_i. Similarly, all the variables in $L(I^q)$ should be assigned to their maximum value, u_i. These are, in both cases, the domain values closest to the mean, which, as noted, lies in the interval I^q. All the remaining unassigned variables in $M(I^q)$ must be assigned the value, v, to satisfy the mean, after all the x_i variables in both $R(I^q)$ and $L(I^q)$ are assigned to their appropriate extreme value. This reasoning means that we need to satisfy $mv + ES(I^q) = q$. Therefore we have $v = \frac{q - ES(I^q)}{m}$.

We can locate the interval I^q such that $V(I^q)$ contains q and classify all the x_i variables to the sets $R(I^q)$, $L(I^q)$, and $M(I^q)$. We restrict our analysis to $R(I^q)$, as the derivation for $M(I^q)$ can be reduced to the same procedure as for the case $R(I^q)$, and the derivation for $L(I^q)$ is symmetric.

Pruning $u_i \; \forall \; x_i \in R(I^q)$. The pruning of x_i has to respect the current upper bound on the standard deviation variable. Let σ_{shift} be the standard deviation that results from adding d to l_i, where $d > 0$. Let d_{max} be the shift amount in l_i such that $\sigma_{shift} = \sigma_{ub}$. Any shift larger than d_{max} will result in $\sigma_{shift} > \sigma_{ub}$, which renders the assignment inconsistent. Therefore, we can prune x_i using $x_i = [l_i, min(l_i + d_{max}, u_i)]$. Shifting l_i by d shifts v to $\frac{q - (ES(I^q) + d)}{m} = v - \frac{d}{m}$. Therefore, $n\sigma^2_{shift}$ is a function of d that can be written as follows:

$$n\sigma^2_{shift}(d) = d^2(1 + \frac{1}{m}) + 2d(l_i - v) + n\sigma^2. \tag{2.9}$$

To solve for d_{max}, let Equation 2.9 equal to $n\sigma_{ub}^2$ and solve for d, we have

$$d_{max} = -b' + \frac{\sqrt{b'^2 - ac}}{a}, \tag{2.10}$$

where $a = 1 + \frac{1}{m}$, $b = 2(l_i - v)$, $c = n\sigma^2 - n\sigma_{ub}^2$, $b' = \frac{b}{2}$.

Note that if $d_{max} > q - \min V(I^q)$, the v value does not reside in I^q. In this case, d_{max} is not valid and it needs to be calculated recursively through the intervals. We refer to Schaus [137, 138] for further details.

2.3.2.4 General Quadratic Constraints

A quadratic constraint represented in the algebraic form can be modelled in CP solvers using the standard form:

$$lower\_bound \leq linear\_expr + nonlinear\_expr \leq upper\_bound.$$

A quadratic objective has the same form but without the bounds. The linear expression *linear_expr* terms are stored in a data structure such as a sparse coefficient list. Each *nonlinear_expr* is represented using an expression tree, where the internal nodes represents operators or functions and the leaf nodes denote variables or constants [59].

We first define the general form of a quadratic constraint. The quadratic constraint is used to reason about constraints with quadratic terms and consists of a set of n variables $\{x_1, \ldots, x_n\}$, a $n \times n$ symmetric matrix $\boldsymbol{H}$, a n-dimensional vector $\boldsymbol{f}$, and scalars l and u. The representation is given as follows:

$$\text{QUADRATIC}(\{x_1, \ldots, x_n\}, \boldsymbol{H}, \boldsymbol{f}, l, u),$$

where $\boldsymbol{H} \in \mathbb{Q}^{n \times n}$, $\boldsymbol{f} \in \mathbb{Q}^n$, $l \in \mathbb{Q}$ and $u \in \mathbb{Q}$. The constraint ensures the following condition:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} H_{i,j}x_i x_j + \sum_{i=1}^{n} f_i x_i \in [l, u].$$

We note that in this general form, the matrix $\boldsymbol{H}$ is not required to be positive semi-definite, i.e., the quadratic constraint need not be convex.

There are two types of global constraints in the literature that reason about quadratic terms in a function. Both can be applied to convex and nonconvex quadratic constraints. We note that although these two filtering algorithms have been tested in the authors' own testing environments, we have not found them implemented in any major CP solver.

Constraint 1. The first global constraint was proposed by Domes and Neumaier [52]. The key idea lies in eliminating bilinear entries in a quadratic constraint and solving the resulting separable quadratic constraints by means of a sequence of univariate quadratic problems.

Without loss of generality, given a univariate quadratic function $a_i x_i^2 + b_i$, we first find the upper bound of the function:

$$u^{'} = \max\{a_i x_i^2 + b_i x_i \mid l_i \leq x_i \leq u_i\},$$

which can be computed with the relationship $u^{'} = \max\{(l_i(a_i l_i b_i)), (l_i(a_i l_i b_i))\}$, as $u^{'}$ must take its maximum value at the boundary of the quadratic function. The lower bound $l^{'}$ can be found by searching

for a value v such that $a_i x_i^2 + b_i$ is minimum: $\frac{\partial a_i x_i^2 + b_i}{\partial x_i}|_{x_i=v} = 0$. If v does not reside within $[l_i, u_i]$, we can set $l^{'} = \min\{(l_i(a_i l_i b_i)), (l_i(a_i l_i b_i))\}$. The new lower and upper bounds are therefore the intersection of the original bounds $[l, u]$ and the bounds due to the quadratic constraint, $[l^{'}, u^{'}]$.

Second, we reduce the domain for x_i by finding its new lower and upper bounds as follows:

$$[l_i^{'}, u_i^{'}] = \{x_i \geq 0 \mid a_i x_i^2 + 2 b_i x_i \geq c\},$$

which can be computed analytically. We can then tighten $[l^{'}, u^{'}]$ to $[l^{'}, u^{'}] \cap [l, u]$.

The results from univariate quadratic functions are then generalized to multi-variable separable quadratic functions of the form:

$$\sum_{i=1}^{n} p_i(x_i) \geq c, \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},$$

where

$$p_i(x_i) = a_i x_i^2 + b_i x_i.$$

The above formulation implies that none of the quadratic terms involves bilinear entries entries, $x_i x_j$, where $i \neq j$. In order to apply the propagation algorithm to a separable quadratic function, the bilinear entries are computed approximately and eliminated from the function so that the resulting quadratic function is separable. Finally, the new lower and upper bounds due to the separable quadratic function are used to infer domain reductions for the x_i variables.

The filtering algorithm was compared to a traditional approach, the elementary constraint propagation, which simply finds the interval of each expression in the quadratic constraint individually, then uses the intervals of all other expressions but itself to compute the new bounds. Results on randomly generated problems show that the proposed filtering algorithm can infer many more domain reductions than the elementary constraint propagation method.

Constraint 2. The second global constraint was proposed by Lebbah et al. [96]. The main idea is to obtain tight linear outer-approximations of the original quadratic constraints, and then use an LP solver to reduce the domain of each variable, by minimizing and maximizing each variable with respect to the linear outer-approximations. Specifically, let the original constraint set be $C = \{g_k(\boldsymbol{x}) \leq 0\}$ that is defined on the quadratic constraints g_k and a set of variables $\boldsymbol{x}$, and let the linear outer-approximation of C obtained with the reformulation-linearization techniques (RLT) [143] be $C^{\mathrm{OA}} = \{d_k(\boldsymbol{x}^{\mathrm{OA}}) \leq 0\}$ that is defined on the linear constraints d_k, where $\boldsymbol{x}^{\mathrm{OA}} = (\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{y}$ are the newly introduced variables required by the approximation technique. The filtering algorithm first finds the new lower and upper bounds for $\boldsymbol{x}^{\mathrm{OA}}$ subject to the constraint set C^{OA} and the current bounds of the variables as follows:

$$l_j^{\mathrm{OA}} = \min x_j^{\mathrm{OA}} \quad \text{subject to } C^{\mathrm{OA}},\ \boldsymbol{l} \leq \boldsymbol{x}^{\mathrm{OA}} \leq \boldsymbol{u}, \tag{2.11}$$

$$u_j^{\mathrm{OA}} = \max x_j^{\mathrm{OA}} \quad \text{subject to } C^{\mathrm{OA}},\ \boldsymbol{l} \leq \boldsymbol{x}^{\mathrm{OA}} \leq \boldsymbol{u}. \tag{2.12}$$

The problems (2.11) and (2.12) can be solved with LP solvers such as CPLEX.

The new bounds $\boldsymbol{l}^{\mathrm{OA}}$ and $\boldsymbol{u}^{\mathrm{OA}}$ are then used to tighten the linear constraints in C^{OA}, and the problems (2.11) and (2.12) are solved again to infer new domain reductions. The process between improving and solving the problems (2.11) and (2.12) ends when no more significant reductions can be made or when

the domain of a variable becomes empty. It was proved that this process terminates in finite number of iterations [96].

A major contribution of this work is the rigorous derivation of the linear outer-approximations. Special care is taken to ensure that the linearization and the Simplex algorithm used by the LP solver do not create any numerical issues that may remove a feasible solution.

The filtering strategy has been tested on the benchmark instances in robotics and kinematics. Results show that the filtering algorithm is able to significantly tighten the initial domains of the variables and finds all the feasible solutions quickly when incorporated within a tree search [96].

We note that although the authors did not make the connection to the MINLP literature, this approach is similar to the optimization-based bound tightening (OBBT) technique [64]. OBBT based approaches are often only done at the root node of the search tree as it is too expensive to perform at every node.

2.3.2.5 Search

Similar to MIP, CP uses a tree search framework. At each node during the search, a variable that has not been assigned a value is first selected with respect to some variable ordering heuristic, then a value is selected, forming the next node. The common branching scheme of CP creates two branches: Let x_i be the variable selected at a node and let v_i be some value in the domain of x_i, then two branches, $x_i = v_i$ and $x_i \neq v_i$, are created. Since the search is basically an enumeration of all the possible variable-value combinations, constraint propagation is applied at each node to reduce the domains of the variables, pruning subtrees containing no solutions. The search continues until a feasible solution is found or until the search proves that no feasible solution exists for a CSP. In case of a COP, in order to prove optimality, a variable, obj, is typically used to represent the objective value, which is linked to the actual objective function with a constraint. When an incumbent is found with the objective value v_{obj} during the search, a constraint of the form $obj < v_{obj}$ (in case of a minimization problem) is added to the problem, constraining the next solution to be better than the current one. The search terminates when no better feasible solution exists, proving the latest solution optimal. The basic CP search algorithm for a CSP is given in Algorithm 2.

Algorithm 2 Basic CP Search Algorithm for a CSP

```
1: CPSearch(Problem P)
2: if propagate(P) → dead-end then
3:    return no-alternative
4: end if
5: if not all variables are fixed then
6:    x_i = choose an unassigned variable in P
7:    v_i = choose a value for x_i from the domain of x_i
8:    if CPSearch(P ∪ (x_i = v_i)) → solution then
9:       return solution
10:   else
11:      return CPSearch(P ∪ (x_i ≠ v_i))
12:   end if
13: end if
```

In addition to the basic search and propagate framework, many advanced techniques have been developed to further improve the performance of CP. The most notable techniques include constraint-based

local search [149], randomization and restart strategies [67, 68], no-good learning [83, 132], backjumping [51], symmetry breaking [61], and sophisticated branching heuristics [123]. The combination of the above techniques enable CP to solve challenging real world applications and outperform MIP on a variety of problems [15].

2.3.3 Constraint Integer Programming

Constraint Integer Programming (CIP) [4] is a generalization of MILP that allows the inclusion of arbitrary constraints, with the restriction that after the integer variables are fixed, the remaining subproblem is a linear program. For clarity, it is useful to introduce the formal definition of CIP [4] as follows:
A *constraint integer program* CIP $= (\mathfrak{C}, I, \boldsymbol{c})$ consists of solving

$$\boldsymbol{c}^\star = \min\{\boldsymbol{c}^T\boldsymbol{x} \mid \mathfrak{C}(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^n,\ x_j \in \mathbb{Z}, \forall j \in I\}$$

with a finite set $\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ of constraints $\mathcal{C}_i : \mathbb{R}^n \to \{0, 1\}$, $i = \{1, \ldots, m\}$, the index set $I \subseteq N = \{1, \ldots, n\}$ of the integer variables, and an objective function vector $\boldsymbol{c} \in \mathbb{R}^n$. A CIP has to fulfill the following condition:

$$\forall \hat{\boldsymbol{x}}_I \in \mathbb{Z}^I\ \exists(\boldsymbol{A}', \boldsymbol{b}') : \quad \{\boldsymbol{x}_C \in \mathbb{R}^C \mid \mathfrak{C}(\hat{\boldsymbol{x}}_I, \boldsymbol{c}_C)\} = \{\boldsymbol{c}_C \in \mathbb{R}^C \mid \boldsymbol{A}'\boldsymbol{x}_C \leq \boldsymbol{b}'\}$$

with $C := N \setminus I$, $\boldsymbol{A}' \in \mathbb{R}^{k \times C}$, and $\boldsymbol{b}' \in \mathbb{R}^k$ for some $k \in \mathbb{Z}_{\geq 0}$.

We note that the linearity restriction of the objective function does not hinder CIP from solving problems with non-linear objective functions, as we can introduce an auxiliary objective variable z to link to the actual non-linear objective function with the relationship $z = f(x)$.

CIP has been extended to solve MIQCPs. In this case, it is required that the subproblem with all integer variables fixed to be a quadratically constrained program [23, 155].

2.3.3.1 CIP Solving Framework

We now describe the main CIP solving process. CIP integrates MIP and CP in a single search tree, where the search is a branching process as in MIP. Each node in the search tree is first treated by CP techniques such as domain propagation. The LP relaxation is then solved and strengthened by MIP techniques such as cutting plane generation. If the bounds of the variables are tightened during the node processing, domain propagation is triggered again to infer further domain reductions. The loop between domain propagation and LP relaxation solving terminates when no more improvement can be made by either technique. Then the search continues depending on the outcome of solving the LP relaxation, i.e., branching is performed if the LP relaxation is feasible, and, in the infeasible case, conflict analysis is performed to learn no-goods [2]. The basic framework of CIP is presented in Fig. 2.2.

In the CIP solver, SCIP, the semantics and algorithms for processing constraints of a specific type are defined in a constraint handler, which acts as an augmented global constraint that embeds both inference techniques in the form of bounds propagation and relaxation-based reasoning via constraint-specific linear relaxations and cutting planes. These techniques are functional extensions to global constraints beyond filtering. We describe the constraint handler for quadratic constraints in the following section.

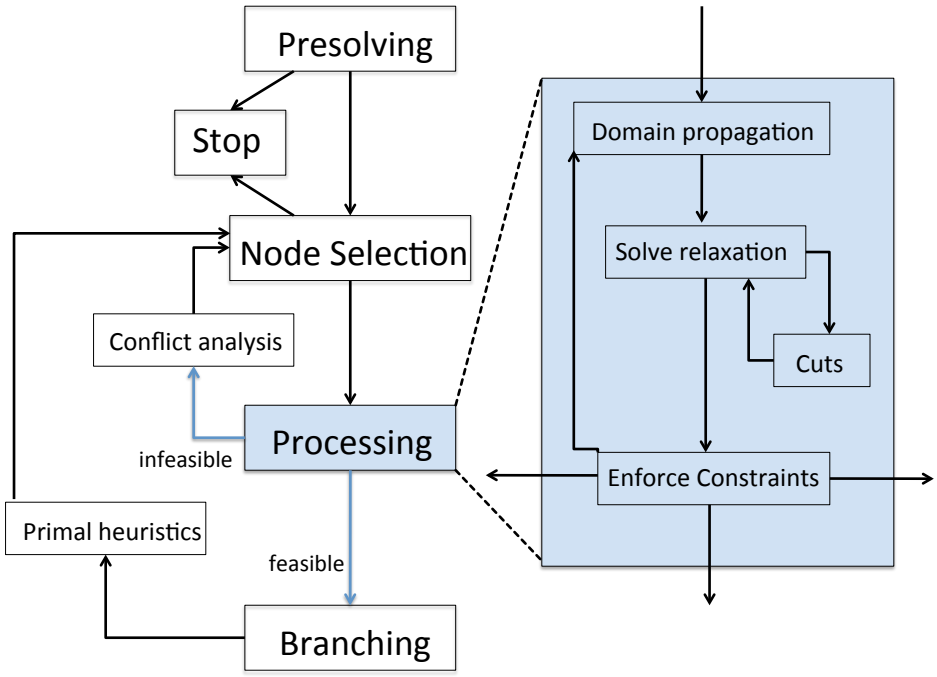


Figure 2.2: The CIP framework. This diagram is taken from Vigerske & Gleixner [155] with slight modification. In "enforce constraints", the algorithm has to decide whether the optimal relaxed solution satisfies all of its constraints. If it is infeasible, the algorithm can try to resolve the infeasibility by adding cutting planes, performing domain reductions, or triggering conflict analysis.

2.3.3.2 The Quadratic Constraint Handler

The quadratic constraint in SCIP follows the same definition of the QUADRATIC constraint in Section 2.3.2.4. The full quadratic constraint handler in SCIP implements a number of problem solving techniques in addition to the propagation algorithm. Computational results show that SCIP with the quadratic constraint handler is competitive to commercial MIP and MINLP solvers on various MIQCP benchmarks and applications [22, 23, 28]. We now explain the functionalities of the quadratic constraint handler using Fig. 2.2.

Presoving. In presolving, SCIP applies two reformulation strategies to quadratic terms with binary variables that reduces the number of quadratic terms in a constraint. First, the square of a binary variable is replaced by the binary variable itself through the relationship $x_i^2 = x_i$. Second, if a constraint contains quadratic terms that are a product of a linear term and a binary variable, i.e., $x \sum_{i=1}^{k} a_i y_i$, where x is binary, y_i is a general integer variable, and $a_i \in \mathbb{Q}$, then this product is replaced with the following constraints:

$$y^L x \leq z \leq y^U x,$$

$$\sum_{i=1}^{k} a_i y_i - y^U(1-x) \leq z \leq \sum_{i=1}^{k} a_i y_i - y^L(1-x),$$

where $y^L = \sum_{i=1,a_i>0}^{k} a_i y_i^L + \sum_{i=1,a_i<0}^{k} a_i y_i^U$ and $y^U = \sum_{i=1,a_i>0}^{k} a_i y_i^U + \sum_{i=1,a_i<0}^{k} a_i y_i^L$.

In addition, the constraint handler recognizes quadratic constraints of the form of a second order

cone (SOC), i.e.,

$$\sqrt{\sum_{i=1}^{k} \gamma + (\alpha_i(x_i + \beta_i))^2} \leq \alpha_0 y + \beta_0, \quad \alpha_0 \geq -\beta_0,$$

where $\alpha_i, \beta_i \in \mathbb{Q}, \forall i, \gamma \in \mathbb{Q}_+$. If the current LP solution $(\bar{\boldsymbol{x}}, \bar{y})$ violates some SOC constraint, then valid cutting planes of the following form can be applied:

$$\eta + \frac{1}{\eta} \sum_{i=1}^{k} \alpha_i^2 (\bar{x}_i + \beta_i)(x_i - \bar{x}_i) \leq \alpha_0 y + \beta_0,$$

where $\eta = \sqrt{\sum_{i=1}^{k} \gamma + (\alpha_i(\hat{x}_i + \beta_i))^2}$.

At the end of presolving, a quadratic constraint is checked for convexity by computing the sign of the minimum eigenvalue of the coefficient matrix $\boldsymbol{H}$. The constraint handler then generates appropriate cutting planes during the search depending on the convexity of the quadratic constraint.

Processing. In Domain Propagation, SCIP implements filtering algorithms that perform domain reductions both on the bounds of the variables, $\boldsymbol{x}$, and the lower and upper bound of the quadratic expression, l and u. Specifically, the quadratic constraint is first rewritten to separate the linear terms from the quadratic terms as follows:

$$\sum_{j \in J} d_j x_j + \sum_{k \in K} (e_k + p_{k,k} x_k + \sum_{r \in K} p_{k,r} x_r) x_k \in [l, u], \tag{2.13}$$

where $J \cup K \subseteq N$, $J \cap N = \emptyset$, and $p_{k,r} = 0$ for $k > r$. We further define the interval

$$q(a, [b^L, b^U], y) := \{by + ay^2 \mid y \in [y^L, y^U], b \in [b^L, b^U]\}, \tag{2.14}$$

which can be computed analytically. To compute the new bounds $[l', u']$, we replace the variables in the linear terms, x_j, and the bilinear terms, x_r, with their interval domain and rewrite (2.13) as follows:

$$\sum_{j \in J} d_k [x_j^L x_j^U] + \sum_{k \in K} ([f_k^L, f_k^U] x_k + p_{k,k} x_k^2) \in [l, u], \tag{2.15}$$

where $[f_k^L, f_k^U] := [e_k, e_k] + \sum_{r \in K} p_{k,r} [x_j^L x_j^U]$. The formulation in (2.15) can be rewritten with the interval defined in (2.14) as

$$[l', u'] := \sum_{j \in J} d_k [x_j^L x_j^U] + \sum_{k \in K} q(p_{k,k}, [f_k^L, f_k^U], x_k)),$$

where $[l', u']$ are the new bounds. If $[l', u'] \cap [l, u] = \emptyset$ then the quadratic constraint cannot be satisfied and the current node can be cut off. Otherwise, we can tighten $[l', u']$ to $[l', u'] \cap [l, u]$. The new bounds on the objective value are then used to tighten the domains of the variables.

During processing, if the current optimal LP solution, $\bar{\boldsymbol{x}}$, violates a quadratic constraint, the constraint handler may add valid cutting planes to strengthen the formulation. In case of a violated convex quadratic constraint, it is possible to linearize the constraint at $\bar{\boldsymbol{x}}$ and add the valid cutting plane to cut

off $\bar{\boldsymbol{x}}$ using the following form:

$$-u - \bar{\boldsymbol{x}}^\top \boldsymbol{H}\bar{\boldsymbol{x}} + (\boldsymbol{f}^\top + 2\bar{\boldsymbol{x}}^\top \boldsymbol{H})\boldsymbol{x} \leq 0.$$

For nonconvex quadratic constraints, valid cutting planes may also be added with underestimators such as the McCormick underestimators [109]. In this dissertation, we restrict our focus on the convex case. For more details, please see [23].

Primal Heuristics. Two primal heuristics have been developed for finding high quality feasible solutions for the MIQCP based on the concept of large neighbourhood search [49]. The key idea is to restrict the search to finding good solutions in a neighborhood of some optimal or feasible solutions. The hope is that such a restriction makes the resulting problem much easier to solve but still maintains the flexibility to find high quality solutions.

The first heuristic is referred to as the QCP local search. The neighborhood is constructed based on a feasible solution, $\hat{\boldsymbol{x}}$, of the MILP relaxation of a MIQCP. If a MILP primal heuristic finds such a solution but it violates some quadratic constraints, a QCP relaxation of the MIQCP is solved with the integer variables fixed to the values of those in $\hat{\boldsymbol{x}}$ of the MILP relaxation. A feasible solution to this QCP is therefore a feasible solution to the MIQCP.

The second heuristic is an extension of the relaxation enforced neighbourhood search [21]. Let $\bar{\boldsymbol{x}}$ be the optimal LP solution at a given node, this heuristic creates a sub-MIQCP problem by fixing all the integer variables which take an integral value in $\bar{\boldsymbol{x}}$ and restricts the bounds of all the integer variables with fractional LP solution value to the two nearest integral values. The sub-MIQCP thus provides a valid solution to the original MIQCP.

2.3.4 Discrete Ellipsoid-based Search

Discrete enumeration based search algorithms originated from solving the unconstrained ILS problem, which is defined as follows.

Unconstrained ILS Problem. Given a real matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with full column rank and a real vector $\boldsymbol{y} \in \mathbb{R}^m$, the unconstrained ILS problem has the following form:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 . \tag{2.16}$$

Lattice theory [6] provides a geometric view of the ILS problem. Let $\boldsymbol{A}$ be the lattice generator matrix, the lattice generated by $\boldsymbol{A}$ is defined as follows:

$$\mathcal{L}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\}.$$

In lattice theory, solving the ILS problem is equivalent to finding the closest point in $\mathcal{L}(\boldsymbol{A})$ to $\boldsymbol{y}$. Therefore, the ILS problem is also referred to as the *closest vector problem* [6].

There are two main approaches of discrete enumeration based search algorithms for solving the ILS problem. The major difference between these two approaches lies in the geometry of the search region which these two approaches explore. The first approach, following the work of Phost [56], enumerates

lattice points, e.g., integer solutions, based on the hyper ellipsoid defined by the objective function, while the other approach, following the work of Kannan [82], enumerates solutions w.r.t. the convex polytope of the objective function. In practice, Phost's method is more commonly used due to its superior search efficiency, while Kannan's method is more often studied for its theoretical value [161]. Schnorr and Euchner [140] made a major improvement upon Phost's method by instantiating the variables in a more efficient order and their algorithm has been considered the state-of-the-art enumeration based search algorithm.

Schnorr and Euchner's method has been successfully applied to communication applications such as multiple-input and multiple-output wireless communications [108], signal processing [73], and GPS positioning [146]. For example, to achieve a very high precision in a global navigation satellite system, an ILS problem is solved to estimate the double-difference carrier phase ambiguities obtained from pairs of satellite and receivers [146]. Loosely, this problem is to find the integer number of wavelengths between the satellite and the receiver in order to precisely estimate their range. Depending on the specific application that is solved with Schnorr and Euchner's method, the search process is sometimes referred to as *sphere decoding* [108] or *least squares ambiguity decorrelation adjustment* [146]. As there does not seem to be a standard name in the communications literature for the search process proposed by Schnorr and Euchner, we have adopted the term *discrete ellipsoid-based search* (DEBS) in this dissertation.

DEBS has been formulated to solve only three types of ILS problems: unconstrained, box-constrained, and ellipsoid-constrained [42, 43, 89, 92]. The unconstrained problem is defined above in (2.16). We define the box-constrained, and ellipsoid-constrained problems as follows.

Box-constrained ILS Problem. The box-constrained ILS problem can be defined as follows:

$$\min_{\boldsymbol{x}\in\mathcal{B}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{2.17}$$

$$\mathcal{B}=\{\boldsymbol{x}\in\mathbb{Z}^n : \boldsymbol{l}\le\boldsymbol{x}\le\boldsymbol{u},\ \boldsymbol{l}\in\mathbb{Z}^n,\ \boldsymbol{u}\in\mathbb{Z}^n\}.$$

Ellipsoid-constrained ILS Problem. The ellipsoid-constrained problem can be defined as follows:

$$\min_{\boldsymbol{x}\in\mathcal{E}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{E}=\{\boldsymbol{x}\in\mathbb{Z}^n : \|\boldsymbol{A}\boldsymbol{x}\|_2^2\le\alpha\}, \tag{2.18}$$

where α is a constant.

As one of our contributions, we extended DEBS for ILS problems with general linear constraints in Section 3.3.

Geometrically, the objective function of the ILS problem defines a hyper-ellipsoid with center $\boldsymbol{A}^{-1}\boldsymbol{y}$ as follows:

$$\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2<\beta, \tag{2.19}$$

where β is a constant. The DEBS method systematically searches for the optimal integer solution inside the ellipsoid.

The DEBS method consists of two phases: reduction and search. The former is a preprocessing step that transforms the given ILS problem into one for which the search process is more efficient [43], and search consists of the enumeration of the integer points inside the ellipsoid defined by the objective function [140]. While reduction is done before the search, it is easier to justify the impact of the reduction on search performance after the search has been presented. We, therefore, describe search and then reduction as they apply to the unconstrained ILS problem and then note the extensions required

for the box-constrained and the ellipsoid-constrained ILS problems.

2.3.4.1 Search

Among several search strategies in the literature, the Schnorr & Euchner strategy is usually considered the most efficient [140]. The approach can be defined as follows.

Suppose the optimal solution $\boldsymbol{z}$ satisfies the bound $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta$, or equivalently $\sum_{k=1}^{n}(\bar{y_k} - \sum_{j=k}^{n} r_{kj}z_j)^2 < \beta$, where β is a constant which can be obtained for any feasible integer solution and $\boldsymbol{R}$ is the matrix $\boldsymbol{A}$ after transformation by the reduction phase (see Equation (2.23) below).

Define the so far unknown (apart from c_n) and usually non-integer variables:

$$c_n = \bar{y}_n/r_{nn}, \quad c_k = (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj}z_j)/r_{kk}, \quad k = n-1, \ldots, 1. \tag{2.20}$$

Note that c_k is a function of z_{k+1} to z_n, and it is fixed when z_{k+1} to z_n are fixed. The above equation can be rewritten as

$$\sum_{k=1}^{n} r_{kk}^2(z_k - c_k)^2 < \beta,$$

which implies the following n inequalities:

$$\begin{aligned}
\text{level } n: &\quad (z_n - c_n)^2 < \frac{\beta}{r_{nn}^2}, \\
\text{level } n-1: &\quad (z_{n-1} - c_{n-1})^2 < \frac{\beta - r_{nn}^2(z_n - c_n)^2}{r_{n-1,n-1}^2}, \\
&\vdots \\
\text{level } k: &\quad (z_k - c_k)^2 < \frac{\beta - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2}{r_{kk}^2}, \\
&\vdots \\
\text{level } 1: &\quad (z_1 - c_1)^2 < \frac{\beta - \sum_{i=2}^{n} r_{ii}^2(z_i - c_i)^2}{r_{11}^2}.
\end{aligned} \tag{2.21}$$

The search starts at level n, heuristically assigning $z_n = \lfloor c_n \rceil$, the nearest integer to c_n. Given the value of z_n, c_{n-1} can be calculated from the above equation as $c_{n-1} = (\bar{y}_{n-1} - r_{n-1,n}z_n)/r_{n-1,n-1}$. From this value, we can set $z_{n-1} = \lfloor c_{n-1} \rceil$ and search continues. During the search process, z_k is determined at level k, where $z_n, z_{n-1}, \ldots, z_{k+1}$ have already been determined, but $z_{k-1}, z_{k-2}, \ldots, z_1$ are still unassigned (note that c_k depends on $z_{k+1}, z_{k+2}, \ldots, z_n$).

At some level $k-1$ in the search, it is likely that the corresponding inequality in (2.21) cannot be satisfied, requiring the search to backtrack to a previous decision. When we backtrack from level $k-1$ to level k, we choose z_k to be the next nearest integer to c_k. More precisely, if there are multiple backtracks to level k, z_k takes on values in the order

$$\lfloor c_k \rceil, \lfloor c_k - 1 \rceil, \lfloor c_k + 1 \rceil, \lfloor c_k - 2 \rceil, \ldots, \quad \text{if } c_k \leq \lfloor c_k \rceil,$$

or

$$\lfloor c_k \rceil, \lfloor c_k + 1 \rceil, \lfloor c_k - 1 \rceil, \lfloor c_k + 2 \rceil, \ldots, \quad \text{if } c_k > \lfloor c_k \rceil.$$

In the Schnorr & Euchner strategy, the initial bound β can be set to ∞. When we find the first integer point, the k-th entry of the first integer point is $\lfloor c_k \rceil$ for $k = n, \ldots, 1$. We can use this integer point to update β, reducing the hyper-ellipsoid. As this is a feasible, but not necessarily optimal solution, after updating β, the algorithm backtracks to search for improving solutions. For the formal definition and treatment of DEBS search, see [140].

The search algorithm for the box-constrained ILS problem is modified to take into account the box-constraints, i.e., $\bar{\boldsymbol{l}} \leq \boldsymbol{z} \leq \bar{\boldsymbol{u}}$, $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$ in (2.25). The modification is done by examining z_k computed at each level, and comparing it with the given variable bounds $\bar{l}_k$ and $\bar{u}_k$ in a straightforward way: During the search, the algorithm keeps track of whether the enumeration of z_k at level k has reached $\bar{l}_k$ and $\bar{u}_k$ from the box-constraint. This mechanism ensures that the search algorithm always enumerates the integers within the box-constraint at each level [43].

Similarly, the algorithm for the ellipsoid-constrained problem has to take into account the bounds imposed by the ellipsoidal constraint [42]. The key idea is to derive the interval defined by the ellipsoidal constraint in (2.24). Rewrite the ellipsoidal constraint $\|\boldsymbol{Rz}\|_2^2 \leq \alpha$ as follows:

$$\sum_{k=1}^{n}(r_{kk}z_k + \sum_{j=k+1}^{n} r_{kj}z_j)^2 \leq \alpha. \tag{2.22}$$

We further define:

$$d_n = \alpha, \quad d_{k-1} = \alpha - \sum_{i=k}^{n}(r_{ii}z_i + \sum_{j=i+1}^{n} r_{ij}z_j)^2 = d_k - (r_{kk}z_k + c_k)^2, \quad k = n, \ldots, 2,$$

where c_k is defined in (2.20). Then inequality (2.22) can be rewritten as follows:

$$(r_{kk}z_k + c_k)^2 \leq d_k, \quad k = n, \ldots, 1.$$

Therefore, z_k is restricted in the interval induced by the ellipsoidal constraint as follows:

$$\left\lceil \frac{-\sqrt{d_k} - c_k}{r_{kk}} \right\rceil \leq z_k \leq \left\lfloor \frac{\sqrt{d_k} - c_k}{r_{kk}} \right\rfloor, \quad k = n, \ldots, 1.$$

We note that the search interval at level k depends on $z_{k+1}, \ldots, z_n$. Therefore, this interval has to be recomputed whenever the search moves from level $k+1$ to k.

In our DEBS implementation, we use an incremental update strategy following the work of Chang [44] for all three types of the ILS problems, so that redundant computation is avoided during the search. Specifically, the partial summation $\sum_{j=l}^{n} r_{kj}z_j$, for some $l > k+1$ may have been already computed when the search previously visited level k. Therefore it is not computed again in the current visit at level k.

2.3.4.2 Reduction

It has been shown that the order of the diagonal entries of $\boldsymbol{R}$ can greatly affect the performance of the DEBS search by reducing the branching factor at the top of the search tree [6, 43]. Note that the n inequalities in (2.21) define an interval of possible values for each variable, z_k, and the smaller this interval, the smaller the number of possible integer values for a given variable. Since search tree size will

tend to be smaller if there is a small branching factor at the top of the tree and the diagonal entries in the $\boldsymbol{R}$ matrix play a large role in determining the size of the interval of possible values for each variable (see (2.21)), the reduction phase of DEBS tries to transform $\boldsymbol{A}$ into an upper triangular matrix $\boldsymbol{R}$ such that the diagonal entries are ordered in non-decreasing order:

$$|r_{11}| \leq |r_{22}| \leq \ldots \leq |r_{kk}| \leq \ldots \leq |r_{n-1,n-1}| \leq |r_{nn}|.$$

Transforming $\boldsymbol{A}$ to $\boldsymbol{R}$ can be achieved by finding an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ and a unimodular matrix $\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$ such that

$$\boldsymbol{Q}^\top \boldsymbol{A}\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_1, \boldsymbol{Q}_2 \end{bmatrix}$$

where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is orthogonal, $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ is upper triangular, and $\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$ is a unimodular matrix. We have

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \left\|\boldsymbol{Q}_1^\top \boldsymbol{y} - \boldsymbol{R}\boldsymbol{Z}^{-1}\boldsymbol{x}\right\|_2^2 + \left\|\boldsymbol{Q}_2^\top \boldsymbol{y}\right\|_2^2,$$

where $\bar{\boldsymbol{y}} = \boldsymbol{Q}_1^\top \boldsymbol{y}$ and $\boldsymbol{z} = \boldsymbol{Z}^{-1}\boldsymbol{x}$.

Therefore, the original ILS problem (4.4.3) is transformed to the new ILS problem:

$$\min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \tag{2.23}$$

where $\boldsymbol{z} = \boldsymbol{Z}^T\boldsymbol{x}$.

The problem (2.23) is exactly the reduced unconstrained ILS problem, on which the search can be carried out directly. For the ellipsoid-constrained problem, the reduced problem can be written as follows:

$$\min_{\boldsymbol{z}\in\bar{\mathcal{E}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \quad \bar{\mathcal{E}} = \{\boldsymbol{z} \in \mathbb{Z}^n : \|\boldsymbol{R}\boldsymbol{z}\|_2^2 \leq \alpha\}. \tag{2.24}$$

In this dissertation, we use the common LLL matrix reduction technique [98] for the unconstrained ILS and ellipsoid constrained problems. Specifically, we implement the partial LLL reduction algorithm proposed by Xie [162], which has lower computational complexity than the original LLL matrix reduction. A comparison of recent LLL reduction strategies is given in Xie [162]. Note that for the ellipsoid-constrained problem, we can still apply the LLL reduction to $\boldsymbol{A}$ as in the unconstrained case, as there is no restriction on the bounds of the variables [42].

For the box-constrained problem, in order to keep the bounds on the variables unchanged, a transformation to $\boldsymbol{A}$ from the right-hand side must be a permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$ instead of a unimodular matrix $\boldsymbol{Z}$. The reduced box-constrained problem can be written as follows:

$$\min_{\boldsymbol{z}\in\bar{\mathcal{B}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \tag{2.25}$$

$$\bar{\mathcal{B}} = \{\boldsymbol{z} \in \mathbb{Z}^n : \bar{\boldsymbol{l}} \leq \boldsymbol{z} \leq \bar{\boldsymbol{u}},\ \bar{\boldsymbol{l}} \in \mathbb{Z}^n,\ \bar{\boldsymbol{u}} \in \mathbb{Z}^n\},$$

where $\bar{\boldsymbol{l}} = \boldsymbol{P}^\top \boldsymbol{l}$ and $\bar{\boldsymbol{u}} = \boldsymbol{P}^\top \boldsymbol{u}$. We use the reduction algorithm due to Chang et al. [43] in this dissertation.

For all three types of the ILS problems, after the optimal solution $\boldsymbol{z}^*$ to a reduced ILS problem is found, the optimal solution $\boldsymbol{x}^*$ to the original ILS problem can be recovered with the relationship $\boldsymbol{x}^* = \boldsymbol{Z}\boldsymbol{z}^*$ for the unconstrained and ellipsoid-constrained problems, and $\boldsymbol{x}^* = \boldsymbol{P}\boldsymbol{z}^*$ for the box-constrained

problem.

As a final note, although DEBS has been successful for solving problems in communications, it has not been well recognized by the OR community. Therefore, as one of our contributions in this dissertation, we study the performance of DEBS on a variety of problems in OR and communications in Chapter 3, and compare its performance with the common approaches in the OR literature.

2.4 Summary

In this chapter we introduced the strictly convex IQCP and reviewed the common exact solution approaches for solving IQCPs. All of the exact approaches described in this chapter are tree-search based enumeration scheme with various techniques to reduce the nodes that need to be explored. This similarity, together with the equivalence between the IQCP and ILS formulations, allows the possibility of efficient hybrid algorithms. In Chapter 3 we develop two hybrid algorithms that integrate MIP and DEBS and CP and DEBS, respectively, for solving some special cases of the IQCP. In Chapter 4, we use the CIP approach to solve the variance minimization problem, an important application of the IQCP. In Chapter 5 we develop a CP approach that combines the techniques from MIP, CP and DEBS for solving general IQCPs.

Chapter 3

Combining DEBS with MIP and CP for Solving IQCPs

3.1 Introduction

As introduced in Chapter 2, mixed integer programming (MIP) and constraint programming (CP) are both general tree search frameworks for solving combinatorial optimization problems and they can be applied to IQCPs. Discrete ellipsoid-based search (DEBS), in contrast, is a specialized tree search algorithm that aims at solving the integer least squares (ILS) problem. In Section 2.2.3 we showed that a mutual reformulation of the IQCP and the ILS problem exists, allowing the potential of solving these two problems with approaches from different fields. Despite the difference in the underlying theories and techniques of these three approaches, abundant information can be extracted from DEBS to enhance several components of MIP and CP for solving IQCPs. In this chapter, we present two hybrid algorithms that, respectively, combine DEBS with MIP and CP.

First, we hybridize DEBS with MIP to solve the strictly convex quadratically-constrained problems with variable bounds, incorporating aspects of DEBS into a presolving technique, cutting planes, and a primal heuristic. While MIP is able to solve problems with general linear constraints, the DEBS method has only been known to solve pure quadratically constrained problems. As test sets, we study the binary quadratic programming (BQP) problem [131] and the unconstrained, box-constrained, and ellipsoid-constrained ILS problems [6]. We perform experiments on benchmark instances for these four problems and compare the performance of two MIP solvers, DEBS, the new hybrid algorithm, and the best known available approaches in the literature. Our experimental results demonstrate that the new hybrid algorithm is competitive to the best known approaches, and, in many cases, establishes the new state of the art.

Second, we hybridize DEBS with CP to solve strictly convex quadratically constrained problems with linear constraints. As noted above, previously DEBS has been formulated to only be applicable to pure quadratically constrained problems. This work is the first that generalizes DEBS to incorporate general linear constraints. From a CP perspective, DEBS can be understood as a specialized CP search that does three things. First, the search strategy implies a fixed variable ordering heuristic that is determined after the reduction phase. Second, DEBS provides a value ordering heuristic. Third, the geometry of the ellipsoid induces an interval domain for each variable. As a result, DEBS is essentially the enumeration

of these domains under the prescribed variable and value orderings. We use this insight to integrate linear constraints into DEBS so that linear constraints are made arc consistent to further reduce the domains of the variables. Although CP has not been known for solving IQCPs efficiently, we show that a combination with DEBS greatly improves the performance of CP. The hybridization is implemented as part of DEBS, but a reverse implementation can also be done: we can implement DEBS as branching heuristics and a global constraint in a CP solver. We turn to this approach in Chapter 5. For test sets, we use the exact quadratic knapsack problem (EQKP), an important class of optimization problem with a number of practical applications. We demonstrate that our hybrid algorithm achieves state-of-the-art for solving the EQKP and a variation.

3.1.1 Contributions

The main contributions of this chapter are as follows:

Hybrid of DEBS and MIP:

- We propose a novel, competitive hybrid algorithm that combines DEBS and MIP and demonstrate it is one of the state-of-the-art approaches for the BQP problems in the operations research (OR) literature. A particular advantage of this hybrid approach is that, unlike the BQP specific approaches such as semi-definite programming (SDP) [87] or the specialized branch and bound (B&B) solver [102], the hybrid algorithm can be applied to a broader class of problems, such as the unconstrained integer quadratically constrained problems and integer quadratically constrained problems with general integer bounds on the variables [90].

- We apply our hybrid algorithm to solve the ILS problems studied in the signal processing literature and demonstrate that it is the state-of-the-art approach. We reformulate the ILS problems as IQCPs and conduct extensive experiments to evaluate our hybrid algorithm on three types of ILS problems: unconstrained, box-constrained, and ellipsoid-constrained problems. We believe such a connection between MIP and DEBS bridges the research fields of OR and communications and may encourage future innovation for solving IQCPs.

- To the best of our knowledge, this work is the first study that applies the DEBS approach to BQP problems. Our experimental results demonstrate that DEBS can outperform state-of-the-art MIP solvers and the best known SDP approaches, especially for smaller problem instances. This suggests that the somewhat different approach from the OR community is worth considering when solving BQP problems.

- To the best of our knowledge, MIP solvers have never been applied to the ILS problems that are of interest to the signal processing community. Our results serve as the first empirical study comparing the DEBS method and MIP solvers for the ILS problems. We identify the key characteristics that are crucial to the performance of each approach. We also show that an off-the-shelf commercial MIP solver performs very well on the box-constrained problems and should be considered as a competitive approach for solving such problems.

Hybrid of DEBS and CP:

- This work is the first time that DEBS has been applied to problems with linear constraints. Our hybridization with CP enables DEBS to solve a much broader class of problems, i.e., problems with a quadratic objective function and any number of linear constraints. We experiment with the EQKP and a variation and achieve the state of the art.
- This work is also the first study that analyzes the DEBS method from a CP perspective. We describe DEBS as a set of branching heuristics and a propagation algorithm.

The work in this chapter is based on two publications [89, 90] and a submitted manuscript [92].

3.1.2 Organization

The rest of the chapter is organized as follows. We present our two hybrid algorithms in two separate sections, Section 3.2 and Section 3.3, organized based on the types of problems that these hybrid algorithms solve. In both sections, we first provide the problem definition and review relevant literature. We then present the hybrid algorithm, computational results, and discussions. We conclude this chapter in Section 3.4.

3.2 Combining DEBS and MIP

In this section we propose a hybrid algorithm that combines DEBS and MIP to solve quadratically-constrained problems with variable bounds. Specifically, we study the BQP problem and the unconstrained, box-constrained, and ellipsoid-constrained ILS problems.

3.2.1 Literature Review

Integer Least Squares (ILS) Problems. ILS problems arise in many signal processing applications such as global positioning system, communications, and cryptography (see e.g., [6, 73, 146]). For example, in the multiple-input and multiple-output wireless communications, the transmitted signal can be estimated via maximum likelihood decoding, or equivalently, by solving an ILS problem [73]. As another example, achieving high precision relative global navigation satellite system positioning [80] requires resolving double differenced carrier phase ambiguities as integers, which again can be naturally formulated as an ILS problem. In the signal processing community, there are four main families of approaches to solving the ILS problems: the Voronoi approach [110], the approximation approach [7], semi-definite programming [85], and DEBS [82] (see Chapter 2 for details). In practice, ILS problems are most commonly solved with DEBS based on enumeration of integer points within the hyper-ellipsoid defining the feasible space.

ILS problems can also be formulated as equivalent IQCPs and solved with branch-and-cut (B&C) based MIP solvers [11] such as CPLEX, GUROBI, and SCIP.

Binary Quadratic Programming (BQP) Problems. BQP problems arise in many combinatorial optimization problems such as task allocation [100], quadratic assignment [57], and max-cut problems [87]. In addition to DEBS, a variety of exact methods exist for solving BQP problems including the linearization method [156], MIP [23, 32], and SDP based B&B approaches [87, 131].

The SDP based approach is often regarded as the state-of-the-art approach for solving BQPs. In this approach, the semi-definite relaxation bound of the objective function is used to prune nodes during the branch-and-bound process. Krislock et al. [87] showed that their SDP algorithm dominates existing approaches for the BQP problems in the Biq-Mac library [158].

As another state-of-the-art approach, Li et al. [102] proposed a specialized branch-and-bound algorithm and demonstrated its strong performance on benchmark instances from a number of sources. Li et al.'s techniques are derived from the geometric structure of the BQP problem based on perturbation analysis. The results of the analysis are implemented in the form of problem-specific lower bounding techniques and inference rules to fix values. Variable and value ordering heuristics, as well as a primal heuristic to find high quality feasible solutions, are also proposed to accelerate the convergence of the search. Empirical results demonstrate that the algorithm is one of the state-of-the-art approaches for finding exact solutions to these benchmark BQP problems.

Mixed Integer Programming for BQP and ILS Problems. MIP is considered the standard generic approach in OR for solving IQCPs exactly [37]. MIP can be further categorized into mixed integer linear programming (MILP) and mixed integer nonlinear programming (MINLP). BQP and ILS are special types of IQCPs, in the sub-class of MINLP that is generally solved with the following four approaches based on MIP-related techniques: branch-and-bound and branch-and-cut (B&C) [70], outer approximation [54], extended cutting-planes [157], and generalized Benders decomposition [62]. Of the above methods, the outer-approximation and the B&C methods have received the most attention and both commercial and non-commercial solvers are available based on these two approaches. An empirical study of these two methods on binary quadratic programming problems concluded that the two approaches performs significantly differently on various classes of problems and it is not clear which is generally superior [32].

The state-of-the-art MIP solvers are typically hybrid algorithms that combine many of the above elements. Modern MIP solvers are able to outperform several other exact approaches [25]. Many solvers such as SCIP [3] and FilMINT [1] incorporate the outer-approximation into the B&C framework and are efficient for solving convex MINLP problems. Commercial software such as CPLEX, GUROBI and BARON also provide functionality to solve convex mixed integer quadratically constrained problems and so can all be applied to solve the BQP problems. We refer interested readers to Bussieck & Vigerske [37] for a review of MINLP solvers.

3.2.2 Problem Definition

3.2.2.1 The Integer Least Squares Problem

The three types of ILS problems are defined as follows.

Unconstrained ILS Problem. Given a real matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with full column rank and a real vector $\boldsymbol{y} \in \mathbb{R}^m$, the unconstrained ILS problem has the following form:

$$\min_{\boldsymbol{x} \in \mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 . \tag{3.1}$$

Box-constrained ILS Problem. The box-constrained ILS problem can be defined as follows

$$\min_{\boldsymbol{x} \in \mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 , \tag{3.2}$$

$$\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n\}.$$

Ellipsoid-constrained ILS Problem. The ellipsoid-constrained problem can be defined as follows

$$\min_{\boldsymbol{x}\in\mathcal{E}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{E} = \{\boldsymbol{x} \in \mathbb{Z}^n : \|\boldsymbol{A}\boldsymbol{x}\|^2 \leq \alpha\}, \tag{3.3}$$

where α is a constant.

3.2.2.2 The BQP Problem

The BQP problem is defined as follows:

$$\min_{\boldsymbol{x}\in\{0,1\}} \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x}, \tag{3.4}$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is a symmetric semi-definite positive matrix and $\boldsymbol{f} \in \mathbb{R}^n$ is a vector.

Note that when $\boldsymbol{H}$ is not symmetric, the symmetric form can be obtained by setting $\boldsymbol{H} = (\boldsymbol{H} + \boldsymbol{H}^\top)/2$. Another common issue for real world applications is the non-convexity property of the matrix $\boldsymbol{H}$. Convexifying $\boldsymbol{H}$ is possible in the BQP case since, for every variable, the relationship $x_i = x_i^2$ holds. Therefore, we can perturb the $\boldsymbol{H}$ matrix and make it semi-definite positive by using a vector $\boldsymbol{u}$ until $(\boldsymbol{H} + \text{diag}(\boldsymbol{u}))$ is semi-definite positive. A computationally inexpensive way to find such a $\boldsymbol{u}$ vector consists of computing the eigenvectors of $\boldsymbol{H}$ [25]. The convex equivalent BQP problem can be obtained as follows:

$$\min_{\boldsymbol{x}\in\{0,1\}} \frac{1}{2}\boldsymbol{x}^\top(\boldsymbol{H} + \text{diag}(\boldsymbol{u}))\boldsymbol{x} + \big(\boldsymbol{f} - \frac{1}{2}\boldsymbol{u}\big)^\top \boldsymbol{x}. \tag{3.5}$$

3.2.3 The Hybrid Algorithm

In this section we describe the three components of DEBS that we integrate into the B&C-based MIP solver SCIP [3]: preconditioning, axis-aligned circumscribed box constraints, and a primal heuristic. Preconditioning applies the DEBS reduction techniques to the IQCP to transform it into a new IQCP problem for which search can potentially be more efficient. The axis-aligned circumscribed box represents globally valid bounds of the variables derived from the geometry of the objective function of the ILS problems. The box is computed after an initial primal solution is found and used to update the bounds whenever a new incumbent solution is found. Finally, we apply the DEBS search as a primal heuristic to find feasible solutions during the B&C search.

3.2.3.1 Preconditioning

Applying the reduction technique from the DEBS method to the IQCP has been shown to be beneficial to the efficiency of the B&C process [11]. The new IQCP problem after performing preconditioning is defined as follows:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \frac{1}{2}\boldsymbol{x}^\top \bar{\boldsymbol{H}}\boldsymbol{x} + \bar{\boldsymbol{f}}^\top \boldsymbol{x}, \tag{3.6}$$

where $\bar{\boldsymbol{H}} = \boldsymbol{R}^\top\boldsymbol{R}$ and $\bar{\boldsymbol{f}} = -\bar{\boldsymbol{y}}^\top\boldsymbol{R}$. Note that $\boldsymbol{R}$ and $\bar{\boldsymbol{y}}$ are from the reduced ILS problem (2.23) in Section 2.3.4.2.

Although the transformation is the same as the reduction in the DEBS method, the reason for the improvement in the B&C search is different. Therefore, we use the word *preconditioning* when it is

applied to the IQCP to distinguish it from the reduction in the DEBS method.

In the context of DEBS, the reduction phase decreases the branching factor early in the search tree as described in Section 2.3.4.2. In contrast, the motivation for performing preconditioning is to reduce the *distance to integrality*. Let $\boldsymbol{x}^{\mathrm{R}}$ denote the optimal solution to the continuous relaxation of the ILS problem and $\boldsymbol{x}^{\mathrm{I}}$ the solution to the ILS problem. We define the distance to integrality as the Euclidean distance between the real and integer optimal solutions to the ILS problem:

$$d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}}) = \left\| \boldsymbol{x}^{\mathrm{R}} - \boldsymbol{x}^{\mathrm{I}} \right\|_2 .$$

Previous results on communication applications showed that the experimental correlation between the number of nodes in the B&C search tree and the distance to integrality in logarithmic scales is around 0.7 [11]. The intuition is that if the root node $\boldsymbol{x}^{\mathrm{R}}$ is closer to the leaf node $\boldsymbol{x}^{\mathrm{I}}$, fewer branching decisions are required and hence the search tree will be smaller. Preconditioning, therefore, aims at decreasing this distance. The relationship between preconditioning and the distance between the real and integer optimal solutions to the ILS problem can be understood as follows. Using $\boldsymbol{A}^{\dagger}$, the Moore-Penrose generalized inverse of $\boldsymbol{A}$, we can write $\boldsymbol{x}^{\mathrm{R}} = \boldsymbol{A}^{\dagger}\boldsymbol{y}$ and $\boldsymbol{A}^{\dagger}\boldsymbol{A} = \boldsymbol{I}$. Therefore

$$d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}}) = \|\boldsymbol{x}^{\mathrm{R}} - \boldsymbol{x}^{\mathrm{I}}\|_2 = \|\boldsymbol{A}^{\dagger}\boldsymbol{y} - \boldsymbol{x}^{\mathrm{I}}\|_2 = \|\boldsymbol{A}^{\dagger}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}})\|_2,$$

leading to

$$d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}}) \leq \|\boldsymbol{A}^{\dagger}\|_2 \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}}\|_2. \tag{3.7}$$

Therefore, it is possible to decrease the distance to integrality by transforming the problem using a unimodular matrix $\boldsymbol{U}$ that minimizes $\|(\boldsymbol{A}\boldsymbol{U})^{\dagger}\|_2$. Here $\boldsymbol{U}$ is obtained by performing the LLL reduction on $\boldsymbol{A}$, which is subjected to the QR decomposition and a series of permutations on the columns of $\boldsymbol{A}$. The details are given in Anjos et al. [11].

We apply the preconditioning technique to all three types of the ILS problems and the BQP problem. In our implementation, we perform the preconditioning in the presolving stage of the B&C algorithm.

3.2.3.2 Axis-aligned Circumscribed Box

Based on the geometry of the objective function of the ILS problem, the axis-aligned circumscribed box can be computed and used to tighten the bounds of the variables. Chang & Golub [42] proposed an efficient way to compute the smallest hyper-rectangle whose edges are parallel to the axes of the coordinate system and that includes the hyper-ellipsoid defined by Expression (2.23). The lower bound $\boldsymbol{l}$ and the upper bound $\boldsymbol{u}$ of the smallest hyper-rectangle including the reduced hyper-ellipsoid $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 \leq \beta$ can be computed by solving the following problems:

$$l_k = \min_{\boldsymbol{z} \in \mathbb{R}^n} \boldsymbol{e}_k^{\top}\boldsymbol{z} \quad \text{subject to } \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 \leq \beta, \tag{3.8}$$

$$u_k = \max_{\boldsymbol{z} \in \mathbb{R}^n} \boldsymbol{e}_k^{\top}\boldsymbol{z} \quad \text{subject to } \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 \leq \beta, \tag{3.9}$$

where $\boldsymbol{e}_k \in \mathbb{Z}^n$ is the unit vector in the k-th direction, i.e., the k-th column of an identity matrix with size n.

Since the variables can only take integer values, we can safely round down the upper bound values

and round up the lower bound values after solving the problems, which gives us:

$$u_k = \left\lfloor \sqrt{\beta} \left\| \boldsymbol{R}^{-\top} \boldsymbol{e}_k \right\|_2 + \boldsymbol{e}_k^\top \boldsymbol{R}^{-1} \bar{\boldsymbol{y}} \right\rfloor,$$

$$l_k = \left\lceil -\sqrt{\beta} \left\| \boldsymbol{R}^{-\top} \boldsymbol{e}_k \right\|_2 + \boldsymbol{e}_k^\top \boldsymbol{R}^{-1} \bar{\boldsymbol{y}} \right\rceil.$$

The axis-aligned circumscribed box can be computed for all three types of the ILS problems and the BQP problem. Since this box is globally valid regardless of the branching decisions, the global bounds of the variables are updated if the box is tighter than the original global bounds of the variables. For the ellipsoid-constrained problem, the box is computed for each ellipsoid: the objective function and the constraint. For each variable, the intersection of the two intervals is taken as the variable bounds.

3.2.3.3 DEBS as a Primal Heuristic

Primal heuristics are used to find feasible solutions in state-of-the-art MIP solvers. Good quality feasible solutions are beneficial in a number of ways [3, 23]. First, the feasible solution proves that the problem is feasible and the solution might already be good enough if the user does not require optimality. Second, the feasible solution provides a valid bound for pruning the search tree. Third, a feasible solution can be used for dual fixing or reduction to strengthen the problem formulation, a common technique in MIP solving [3].

When applying DEBS as a primal heuristic, we employ a computational resource bound (e.g., a short time limit) to run DEBS search at the root node after MIP presolving. If the DEBS search finds an optimal solution, it is returned and the algorithm terminates. Otherwise, the best solution found by the DEBS method is used as the starting integer solution. In addition, DEBS search is executed at selected nodes in the B&C search tree. If the solution found by the DEBS is better than the current incumbent, the current incumbent is updated. Specifically, we run the primal heuristic at a tree node whenever a variable is fixed: During the B&C process, variables might be fixed either by branching decisions or other techniques. When such fixing takes place, we can efficiently reduce the size of the problem, so running the DEBS method is much more efficient. Let $\boldsymbol{U}$ be the set of indices of variables that are not yet fixed at a given node in the search tree and let $\boldsymbol{A}_i$ be the i-th column vector of $\boldsymbol{A}$ of the original ILS problem. The updated ILS can be defined as follows:

$$\min_{\tilde{\boldsymbol{x}} \in \{0,1\}} \left\| \tilde{\boldsymbol{y}} - \tilde{\boldsymbol{A}} \tilde{\boldsymbol{x}} \right\|_2^2,$$

where

$$\tilde{\boldsymbol{A}} = [\boldsymbol{A}_i], \quad i \in \boldsymbol{U},$$

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} - \sum_{i \notin \boldsymbol{U}} \boldsymbol{A}_i x_i,$$

$\tilde{\boldsymbol{A}} \in \mathbb{R}^{n \times |\boldsymbol{U}|}$ and $\tilde{\boldsymbol{y}} \in \mathbb{R}^n$. The resulting problem is $(n - |U|)$ dimensional smaller than the original problem.

We summarize our hybrid algorithm with the flowchart in Figure 3.1. The relationship between the flowchart and Algorithm 1 in Section 2.3.1.2 is as follows. The main solving loop in Figure 3.1 consists of the while loop from line 3 to line 20 in Algorithm 1. The preconditioning and the first execution of the primal heuristic are done before the main solving loop, i.e., between line 2 and line 3. During the

main solving loop, the primal heuristic is executed before solving the continuous relaxation problem at line 4, whenever a variable is fixed. When the incumbent solution is updated (line 11), we compute the axis-aligned box and use it to tighten the global bounds of the variables. This is done between line 11 and line 12.

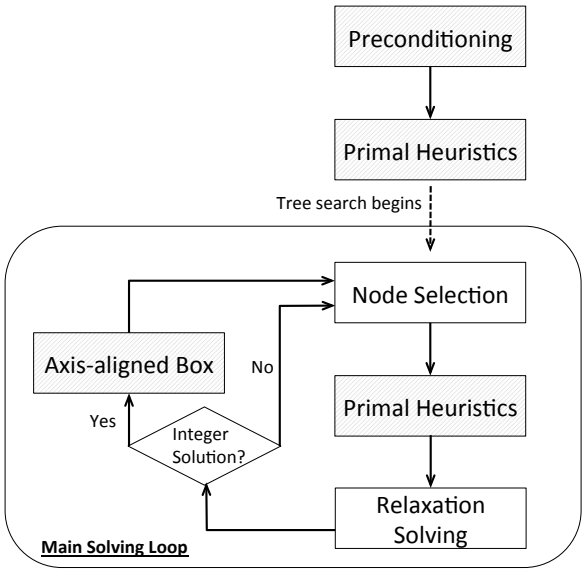


Figure 3.1: The flowchart of the hybrid algorithm. The three components are colored with shades of gray.

In our hybrid implementation, we branch on the variable with the smallest domain and choose the node with the smallest relaxed optimal objective value.

3.2.4 Experimental Results: ILS Problems

3.2.4.1 Setup

All experiments were performed on a Intel Core i7 3.40 GHz machine (in 64 bit mode) with 8GB memory running Red Hat Enterprise 6.2 with one thread. We evaluated three different solution approaches: "MIP": using CPLEX v12.5 as the MIP solver, "DEBS": both the partial LLL reduction and the search are written in C with the GNU Scientific Library (GSL), and "HYBRID": the hybridization of DEBS and the MIP solver SCIP, implemented in SCIP v3.0.2. We use MATLAB 7.7.0 for generating the problem instances. The CPU time limit for each run on each problem instance is 3600 seconds. Since the running times of different approaches for different problem sizes spread across a very wide spectrum, we assume a 0.01 minimum run-time for legibility of the graphs. That is, if the running time of a given approach is less than 0.01 for some instances, we report its running time as 0.01.

3.2.4.2 Test sets

The problem instances for the unconstrained, box-constrained and ellipsoid-constrained problems are generated as in the literature [11, 42, 43]. Each problem instance is generated with the linear model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v}$, where the noise vector $\boldsymbol{v} \in \mathbb{R}^n$ is randomly generated by $\sigma \times$ `randn(n,1)`, the matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is randomly generated by $\boldsymbol{A} =$ `randn(n,n)` and $\boldsymbol{x} \in \mathbb{Z}^n$ is randomly generated in different

ways for each of the three types of ILS problems (see below). In wireless communications, the signal-to-noise ratio is usually used as a measure of the difficulty of the problems [43]. We use $\sigma = 0.01$ as a representative of the low noise range, $\sigma = 0.05$ to represent moderate noise, and $\sigma = 0.5$ for high noise range. Here `randn(n,1)` is a MATLAB function which returns an n-dimensional vector containing pseudorandom values drawn from the standard normal distribution $N(0, 1)$, and `rand(n,1)` is a MATLAB function which returns an n-dimensional vector containing pseudorandom values drawn from the standard uniform distribution on the interval $(0, 1)$, and `round`($\boldsymbol{x}$) is MATLAB function which rounds the elements of $\boldsymbol{x}$ to the nearest integers.

In addition to noise (σ), we also manipulated the range of the initial bounds on the variables (c for the unconstrained and box-constrained problems and α for the ellipsoid constrained problems) as follows.

- Unconstrained Problems: the nine different variations are generated as follows:

$$\boldsymbol{x} = \texttt{round}(c \times \texttt{rand(n,1)}),$$

  where $c = \{3, 10, 100\}$ and $\sigma = \{0.01, 0.05, 0.5\}$.

- Box-constrained Problems: the nine different variations are generated as follows:

$$\boldsymbol{x} = \texttt{round}(c \times \texttt{rand(n,1)}), \mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{c},$$

  where $c = \{3, 10, 100\}$ and $\sigma = \{0.01, 0.05, 0.5\}$. Note that $\boldsymbol{c}$ is a n-dimensional vector with all its entries equal to c.

- Ellipsoid-constrained Problems: the six different variations are generated as follows:

$$\boldsymbol{x} = \texttt{round}(\boldsymbol{lb} + (\boldsymbol{ub} - \boldsymbol{lb}) \times \texttt{rand(n,1)}),$$

$$\text{s.t. } \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \alpha,$$

  where $\alpha = \{0.5n, n\}$, n is the problem size (see below), and $\sigma = \{0.01, 0.05, 0.5\}$. The lower bounds $\boldsymbol{lb}$ and upper bounds $\boldsymbol{ub}$ are the minimal axis-aligned box for the ellipsoidal constraint $\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \alpha$. The $\boldsymbol{x}$ vector is first generated randomly w.r.t. this box. If it is inside the ellipsoidal constraint, we use it. Otherwise we continue generating a new $\boldsymbol{x}$ until it satisfies the ellipsoidal constraint.

For each problem variation, the problem instances consist of ten problem sets with sizes $n \times n$ ranging from 10×10 to 100×100 with a step size of 10. Each set consists of 50 problem instances for a total of 500 instances for each problem variation. The total number of problem instances is therefore $12,000 = 500 \times (9 + 9 + 6)$.

Our motivation for choosing these particular sets of simulations is to analyze the performance of the three approaches with respect to different noise level, problem size, and domain size of the variables.

3.2.4.3 Summary of Results

We summarize the results in Table 3.1, which lists the best approaches for each of the experiments. We use the bootstrap paired-t test [46] with $p \leq 0.01$ to compare the performance of each pair of the

approaches for each of the 24 problem variations. The table shows that the hybrid algorithm outperforms DEBS and MIP overall. It achieves the best performance in 21 of the 24 variations and performs uniquely the best in 3 of them. It performs better or equal compared to DEBS across the 24 variations, including 18 tied with DEBS as the bootstrap paired-t tests indicate that there is no significant difference in average running time for the two approaches. While MIP achieves the best performance in only 3 variations, it is uniquely the best in each of the variations. Note that in two of those three variations, the hybrid approach is significantly better than DEBS.

Unconstrained

		$c = 3$	$c = 10$	$c = 100$
$\sigma = 0.01$	MIP	0.03	0.04	0.03
	DEBS	**0.01**	**0.01**	**0.01**
	HYBRID	**0.01**	**0.01**	**0.01**
$\sigma = 0.05$	MIP	0.06	0.04	0.05
	DEBS	**0.01**	**0.01**	**0.01**
	HYBRID	**0.01**	**0.01**	**0.02**
$\sigma = 0.5$	MIP	23.38	14.51	5.15
	DEBS	56.32	55.80	55.80
	HYBRID	**0.66**	**0.36**	**0.96**

Box-constrained

		$c = 3$	$c = 10$	$c = 100$
$\sigma = 0.01$	MIP	0.04	0.04	0.03
	DEBS	**0.01**	**0.01**	**0.01**
	HYBRID	**0.01**	**0.01**	**0.01**
$\sigma = 0.05$	MIP	0.04	0.04	0.05
	DEBS	**0.01**	**0.01**	**0.01**
	HYBRID	**0.01**	**0.01**	**0.01**
$\sigma = 0.5$	MIP	**0.06**	**0.28**	**2.39**
	DEBS	101.15	196.61	169.16
	HYBRID	0.32	1.08	68.69

Ellipsoid–constrained

		$\alpha = 0.5n$	$\alpha = n$
$\sigma = 0.01$	MIP	3.23	2696.75
	DEBS	**0.01**	**0.01**
	HYBRID	**0.01**	**0.01**
$\sigma = 0.05$	MIP	5.01	2688.88
	DEBS	**0.01**	**0.01**
	HYBRID	**0.01**	**0.01**
$\sigma = 0.5$	MIP	533.45	2649.62
	DEBS	29.47	42.89
	HYBRID	**0.23**	**0.19**

Table 3.1: Mean running time in seconds over all the n values (number of variables) and best approaches for each problem variation. Bold numbers indicate the best approaches for a given problem variation w.r.t. the paired-t test with $p \leq 0.01$.

3.2.4.4 Detailed Results

Before comparing the three solution approaches, we note that SCIP, the solver used in our hybrid algorithm, can also solve pure MIP problems. However, in each problem instance we tested, SCIP is dominated by CPLEX and therefore we only report the MIP results from CPLEX.

In addition, for the unconstrained and the ellipsoid-constrained problems, SCIP often return error messages due to no initial bounds on the variables. Therefore, our hybrid algorithm, i.e., the axis-aligned circumscribed box component, provides finite bounds for all the variables and avoids the issue of unboundedness.

Comparison of the Three Solution Approaches. In the following figures, the average CPU time for finding and proving optimality on the 50 different problem instances versus the dimension of the three types of the problems are presented. In this section, we only present the results that are representative for our discussion and leave the complete results in Appendix A.1.

Unconstrained Problems. For the unconstrained ILS problems, the hybrid algorithm performs the best in general. It is only slightly slower than the DEBS method for problems with small and moderate noise ($\sigma = 0.01$ and $\sigma = 0.05$) due to preprocessing overhead and performs about two orders of magnitude faster on problems with large noise ($\sigma = 0.5$). The hybrid algorithm is able to prove optimality for each instance in around one one-hundredth of a second.

In Fig. 3.2, the results show the results for $c = 10$ as the results for different c values have a very similar relative performance. Comparing to MIP, the hybrid algorithm consistently outperforms MIP for all instances. Comparing the DEBS method to MIP, DEBS greatly outperforms MIP for all problems with small and moderate noise ($\sigma = 0.01$ and $\sigma = 0.05$), proving optimality for all instances in less than one one-hundredth of a second. As the noise becomes larger, all the three approaches are less efficient. However MIP performs better than the DEBS method and the hybrid algorithm remains the best for all tested problem instances.

The results for problems with large noise, in particular, demonstrate the value of hybridizing DEBS and MIP solving. It is not simply that the hybrid approach is able to achieve the minimum run-time (after overhead) of DEBS and MIP. Rather the combination of techniques allows the hybrid to achieve lower run-time on many problem instances than either DEBS or MIP could achieve. A closer examination shows that a problem instance for which DEBS struggles to find good solutions can often be solved to optimality very quickly by DEBS after some variables are restricted during the branch-and-cut process, therefore speeding up the search.

Box-constrained Problems. As with the unconstrained problems, we only show the figure for $c = 10$ because results for different c values are similar. Fig. 3.3 demonstrates that for problems with small and moderate noise ($\sigma = 0.01$ and $\sigma = 0.05$), the DEBS method, MIP and the hybrid algorithm are all able to prove optimality quickly. For problems with large noise ($\sigma = 0.5$), the hybrid approach is approximately two orders of magnitude faster than DEBS but about one order of magnitude slower than MIP. The reason that the hybrid approach underperforms MIP is mainly because the underlying MIP solver included in the hybrid algorithm, SCIP, is not as efficient as CPLEX, which is used to report the MIP running time.

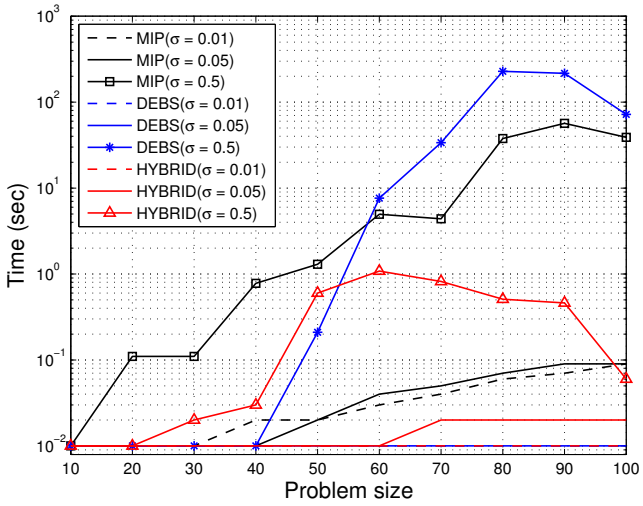


Figure 3.2: Average running time vs dimension for the unconstrained problems, $c = 10$.

For problems with large noise, the DEBS method has great difficulties solving problems with size larger than $n = 50$, while the hybrid algorithm and MIP are still able to solve all the problems. MIP performs the best on all the box-constrained problem instances, specifically, outperforming the DEBS method by about three orders of magnitude when both noise and problem size are large: an off-the-shelf solver defeats the special purpose algorithm for the box-constrained ILS problems.

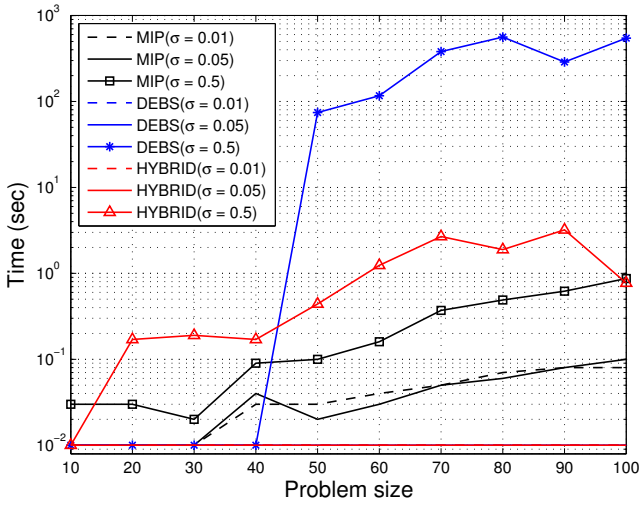


Figure 3.3: Average running time vs dimension on the box-constrained problems, $c = 10$.

Ellipsoid-constrained Problems. The results of the ellipsoid-constrained problems are presented in Figs. 3.4 and 3.5. The hybrid algorithm performs the best, and both the DEBS method and the hybrid algorithm significantly outperform MIP. While the performance for the DEBS method and the hybrid algorithm are similar for both the $\alpha = 0.5n$ and $\alpha = n$ cases, the results for MIP show an interesting difference. MIP is much more efficient for the former. This suggests that MIP performs better when the

constraining ellipsoid is smaller. The reason is likely that the linear approximations are much tighter for the $\alpha = 0.5n$ case. For example, the difference in average optimality gap at the root node for instances with $\sigma = 0.5$ is 5.37% for $\alpha = 0.5n$, and 9.19% for $\alpha = n$.

For large problems with $\alpha = n$ (Fig. 3.5), we again see the synergy arising from the hybridization: the hybrid achieves lower run-time than either DEBS or MIP on many instances of size 50 and greater.

We note that in Fig 3.4, the results of DEBS on the size 60 and size 90 problems are orders of magnitude better than slightly smaller and larger problems. This behaviour is due to the method that is used to generate the problem instances: When $\alpha = 0.5n$, the constraining ellipsoid is already small, which benefits DEBS solving greatly. Therefore most of the instances in this problem set are actually easy to solve with DEBS. Our results show that DEBS solve all of the 50 instances generated for size 60 and size 90 very quickly.

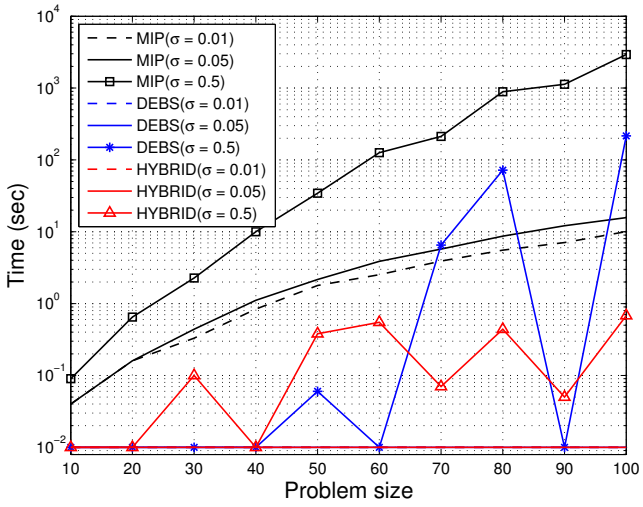


Figure 3.4: Average running time vs dimension on the ellipsoid-constrained problems, $\alpha = 0.5n$.

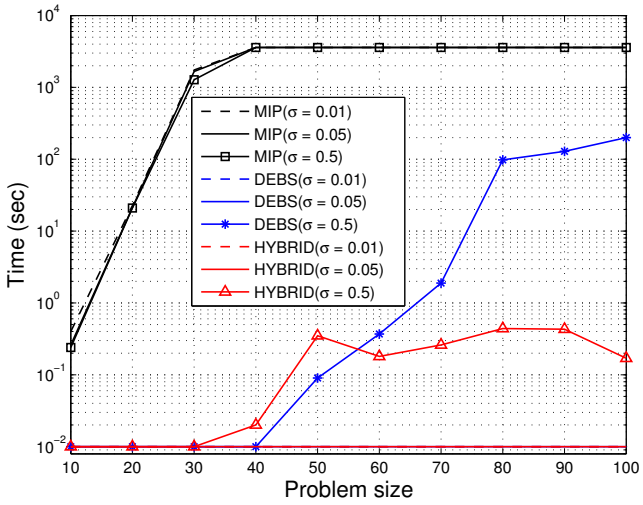


Figure 3.5: Average running time vs dimension on the ellipsoid-constrained problems, $\alpha = n$.

3.2.4.5 Factors that affect the performance

In this section we discuss the impact of our three independent variables: noise, the size of the circumscribed box, and the size of the problem.

Noise. The noise directly affects the size of the ellipsoid defined by the objective function of the ILS problems. A larger noise leads to a larger ellipsoid thus increasing the search space. Recall the ellipsoid with center $\boldsymbol{A}^{-1}\boldsymbol{y}$:

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^*\|_2^2 < \beta,$$

where $\boldsymbol{x}^*$ is the optimal solution and β is a constant. If the noise $\boldsymbol{v}$ is increased, the expression $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^*\|_2^2$ will also increase, thus increasing the volume of the ellipsoid. The increment in noise seems to have a smaller impact on MIP than on the DEBS method. This might be due to the fact that MIP solves the problems by recursively dividing the original problem into smaller subproblems and, with the help of the dual bounding mechanism, MIP is able to cut off more solution space than the DEBS method. Fig. 3.3 demonstrates the impact of different noise levels on the box-constrained problems with $c = 10$, where MIP is less efficient than DEBS with $\sigma = 0.01$ and $\sigma = 0.05$ but significantly outperforms DEBS with $\sigma = 0.5$. For all three types of the ILS problems, the performance of the DEBS method greatly deteriorates from being able to solve all larger problem instances ($n \geq 50$) in less than 0.01 seconds to more than 100 seconds, about five orders of magnitude slower. MIP, on the contrary, exhibits a steadier increase of running time as the noise is increased. It is less efficient than DEBS when noise is small, but is able to outperform DEBS when the noise is large ($\sigma = 0.5$) for the unconstrained and the box-constrained problems. For the ellipsoid-constrained problem, the quadratic constraint greatly increases the difficulty for MIP. Therefore, even though MIP is less sensitive to noise in general, it is outperformed by DEBS regardless of noise level on ellipsoid-constrained problems.

The Size of the Box. Since the outer-approximations in MIP depend heavily on the tightness of the bounds on the variables, MIP performs much better when the range of the initial bounds of the variables is smaller. This is reflected in the results of the unconstrained problems and box-constrained problems. Fig. 3.6 clearly shows the impact of different size of boxes for MIP with $\sigma = 0.5$.

For the DEBS method, we observe that, surprisingly, adding any box constraints is counter-productive. For the difficult problems with large noise $\sigma = 0.5$, Fig. 3.2 and Fig. 3.3 show that the running time for the unconstrained problems is consistently lower than that of the box-constrained problems. The reason is that with bounds on the variables, the DEBS reduction must use permutation matrices instead of the more powerful general unimodular matrices in order to maintain the box-constrained structure without introducing linear constraints, which cannot be handled by the DEBS method. The use of permutation matrices greatly reduces the desired property of the matrix in the DEBS method. Therefore, the DEBS method performs much better for the unconstrained and the ellipsoid-constrained problems, where more powerful reduction techniques are used.

Size of the Problem. Since both MIP and the DEBS method are exact approaches, the search space grows exponentially when the size of the problem is increased. As shown in Figs. 3.2, 3.3, and 3.4, the running time for MIP increases more steadily as problem size becomes larger. In contrast, the performance of the DEBS method deteriorates greatly with problem size larger than $n = 50$. Judging from the results, it seems that the DEBS method is very efficient for all problem sizes when the noise

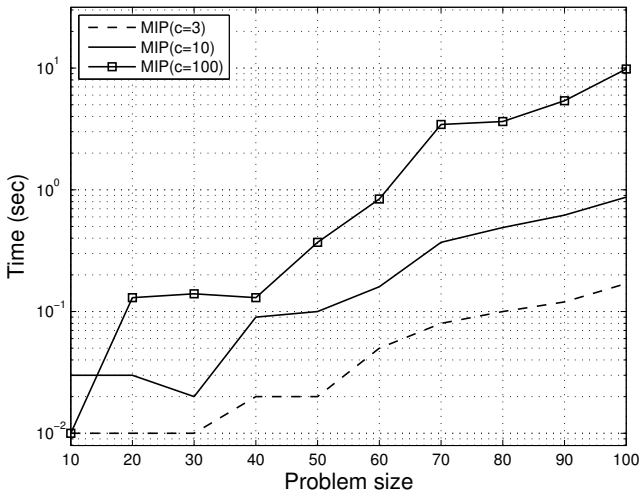


Figure 3.6: Average running time vs dimension for MIP on the box-constrained problems with different initial bounds, $\sigma = 0.5$.

is small or moderate, but fails to scale when the noise is large. When the noise is large, the ellipsoid defined by the objective function is much larger and it becomes extremely difficult to search discretely for the optimal solution.

3.2.5 Experimental Results: BQP Problems

In this section, we turn our attention to the second class of problems: BQPs. As noted in Section 3.2.1, these problems are well studied in the OR literature, in contrast to the problems in the previous section which are studied primarily in the communications literature.

In order to compare our results to the current state-of-the-art, we use the online SDP solver BiqCrunch [88] and we also provide the results on BQPs from Li et al. [102] and Krislock et al. [87].

3.2.5.1 Setup

The environment is the same as the one used for the ILS problems in Section 3.2.4.1.

3.2.5.2 Test sets

We use a subset of the benchmark instances presented by Li et al. [102], excluding the max-cut problems as they require additional transformations and cannot be solved with MIP or DEBS directly. We perform experiments on medium size BQP problems on seven problem sets: Carter type problems [40], William type problems [159], bqp50 and bqp100 problems, and gkaia, gkaib, and gkaid problems [158]. We generate ten problem instances for each of the Carter and William type problems and use the problem instances in the public benchmark sets for the other problem types.

In order to ensure convexity, we perform the same transformation as Li et al. We compute the smallest eigenvalue for the $\boldsymbol{H}$ matrix of each problem and let it be λ_{min}. Then we apply the perturbation vector $\boldsymbol{u} = (-\lambda_{min} + 0.001)\boldsymbol{e}$ to the BQP problem (3.5) when λ_{min} is negative. Note that the transformed problem has the same optimal solution as the original problem. For the DEBS method, we use Cholesky

decomposition on matrix $\boldsymbol{H}$ in the BQP problem to obtain matrix $\boldsymbol{A}$ in the binary ILS problem, and we obtain $\boldsymbol{y}$ from the equation $\boldsymbol{f} = -\boldsymbol{y}^\top \boldsymbol{A}$, which gives us $\boldsymbol{y} = -(\boldsymbol{f}\boldsymbol{A}^{-1})^\top$.

The results for all seven problem sets are presented in Table 3.2. For the bqp50, bqp100, gkaia, gkaib, and gkaid problems, we report the CPU time for each instance. For the Carter and Williams type problems, we report the arithmetic mean CPU time "arith", and the shifted geometric mean CPU time "geo" on the ten instances for each problem size.[1]

3.2.5.3 Summary

Our experimental results demonstrate that DEBS performs much better than CPLEX on average, though approximately at the same level as default SCIP (without the hybrid extensions) on seven standard benchmark sets. Interestingly, DEBS is both significantly better and significantly worse than SCIP on different problem sets. The hybrid approach, implemented in SCIP, outperforms CPLEX, SCIP, and DEBS on all seven problem sets, though on some sets the improvement is marginal. Li et al.'s approach dominates the B&C solvers and DEBS overall while being comparable to our new hybrid approach: on one problem set Li et al.'s approach is superior, on another the hybrid performs approximately an order-of-magnitude better, and on the remaining five sets, the performance of the two algorithms is essentially equivalent. When compared to the SDP approach, the hybrid algorithm performs as efficiently for moderate size problems but lags behind the SDP approach for large problems. However, on some dense problems, the hybrid algorithm greatly outperforms the SDP approach. Overall, the SDP approach still appears to be the strongest for solving BQPs.

3.2.5.4 Detailed Results

The comparison of the identical MIP models in CPLEX and SCIP shows, somewhat surprisingly, that CPLEX performs substantially better than SCIP on only two of the seven problem sets (Williams and Carter). SCIP is clearly superior for bqp50, bqp100, gkaia, and gkaib sets while solving one problem more for gkaid. We attribute this strong performance of SCIP to the quadratic constraint handler described in Berthold et al. [23].

DEBS is noticeably more efficient than B&C (both CPLEX and SCIP) on the William and Carter type problems and superior to CPLEX on the bqp50, gkaia, and gkaib problems. While CPLEX does not out-perform DEBS for any problem instances, SCIP has the edge over DEBS on bqp50, bqp100, and gkaia. For the final problem set, gkaid, SCIP is able to solve one more problem than DEBS. Comparing the DEBS method and CPLEX, the DEBS method is noticeably more efficient than CPLEX for five problem sets: William type, Carter type, bqp50, gkaia and gkaib. Therefore, we conclude that the DEBS approach from the communications literature is a competitive approach to solving BQPs that are of interest to the OR community. In fact, DEBS is superior to a state-of-the-art commercial solver, CPLEX.

Turning to the hybridization of B&C and DEBS, we see that the new hybrid approach out-performs CPLEX on all problem sets while achieving clearly better performance than SCIP on the Williams and Carter type problems and marginally better performance on the other five problem sets: there are a number of problem instances in the latter five sets where the hybrid makes meaningful improvements on the default SCIP performance and none where it is substantially inferior. Similarly, the hybrid improves

[1]The shifted geometric mean time is computed as follows: $\prod(t_i + s)^{1/n} - s$, where t_i is the actual CPU time, n is the number of instances, and s is chosen as 10. Using geometric mean can decrease the influence of the outliers of data [3].

Table 3.2: A comparison of four approaches plus Li et al.'s results and Krislock et al.'s SDP results for the seven problem sets. Bold numbers indicate the best approach for a given problem. The symbol '-' means that no problem instances were solved to optimality within 3600 seconds. For the Carter and William type problems, n is the size of the problem, and p and d are problem generation parameters. The superscripts indicate the number of instances for which no optimal solution was found. We note that Li et al.'s results are due to [102] therefore the run-time information is not consistent with the rest of the results.

Carter type problems

n	p	CPLEX		SCIP		DEBS		Hybrid		Li	SDP
		arith	geo	arith	geo	arith	geo	arith	geo	arith	arith
40	0.2	0.21	0.21	29.55	21.06	**0.01**	0.01	0.02	0.02	0.32	0.32
40	0.3	0.38	0.38	34.01	26.35	**0.01**	0.01	0.03	0.03	0.26	0.62
50	0.2	2.38	2.26	896.53	655.17	**0.15**	0.15	0.24	0.24	1.5	0.66
50	0.3	2.85	2.69	891.71	767.32	**0.80**	0.80	1.10	0.85	1.2	1.15
80	0.4	1170.01^{2}	349.34^{2}	-	-	1020.20^{2}	249.31^{2}	98.37	67.40	89.9	**21.39**
80	0.5	659.99	369.26	-	-	225.33	60.03	48.09	31.02	**2.5**	16.01
100	0.5	3222.90^{8}	2955.28^{8}	-	-	2766.31^{7}	2226.48^{7}	2249.53^{5}	1584.12^{5}	**10.8**	57.97

William type problems

n	d	CPLEX		SCIP		DEBS		Hybrid		Li	SDP
		arith	geo	arith	geo	arith	geo	arith	geo	arith	arith
40	0.5	0.25	0.25	8.29	7.59	**0.01**	0.01	**0.01**	0.01	0.44	0.33
40	0.7	0.22	0.21	14.16	12.16	**0.01**	0.01	**0.01**	0.01	0.47	0.34
50	0.2	1.75	1.70	12.00	10.51	**0.35**	0.32	0.44	0.40	6.6	0.52
50	0.4	1.74	1.68	148.64	115.64	**0.09**	0.09	0.14	0.14	24	0.75
50	1	2.78	2.61	1463.02^{2}	948.47^{2}	**0.13**	0.09	0.19	0.18	2.5	0.85
60	0.1	21.83	15.29	4.61	4.17	1.57	1.51	1.93	1.85	**0.72**	**0.72**
60	0.2	22.81	11.20	387.15	163.29	2.10	1.84	2.65	2.26	5.3	**1.33**
60	0.4	14.18	11.49	2568.63^{4}	1833.75^{4}	31.21	10.48	6.24	5.17	18.9	**2.32**
80	0.1	2014.32^{5}	1047.60^{5}	1688.28^{3}	850.38^{3}	157.52	80.99	178.82	89.52	7	**3.01**
80	0.2	1023.81^{1}	530.40^{1}	-	-	188.92	97.89	140.83	82.79	123.1	**2.19**

bqp50 problems

Instance	CPLEX	SCIP	DEBS	Hybrid	Li	SDP
bqp50-1	11.66	0.07	0.65	**0.06**	<0.2	0.25
bqp50-2	1.84	0.06	1.20	**0.03**	<0.2	0.24
bqp50-3	0.60	0.04	**0.01**	0.03	<0.2	0.17
bqp50-4	2.84	0.05	0.30	**0.04**	<0.2	0.22
bqp50-5	3.72	0.03	0.13	**0.02**	<0.2	0.21
bqp50-6	0.46	**0.01**	0.15	0.02	<0.2	0.21
bqp50-7	1.70	0.05	**0.02**	0.05	<0.2	0.26
bqp50-8	0.93	0.06	**0.03**	0.06	<0.2	0.19
bqp50-9	5.32	0.09	0.35	**0.07**	<0.2	0.25
bqp50-10	14.30	0.07	0.25	**0.06**	<0.2	0.24

bqp100 problems

Instance	CPLEX	SCIP	DEBS	Hybrid	Li	SDP
bqp100-1	-	55.50	-	46.13	1648	**1.72**
bqp100-2	-	6.74	-	7.12	63.4	**2.06**
bqp100-3	-	4.99	-	9.10	66.3	**1.60**
bqp100-4	-	8.83	-	8.96	165.7	**1.42**
bqp100-5	-	10.28	-	9.17	1230	**1.69**
bqp100-6	-	566.92	-	55.54	175.2	**11.12**
bqp100-7	-	42.52	-	13.37	427	**1.80**
bqp100-8	-	4.63	-	5.96	25.6	**1.63**
bqp100-9	-	3.50	1282.42	5.90	21.9	**1.46**
bqp100-10	-	4.43	-	4.45	57.2	**1.67**

gkaia problems

Instance	n	CPLEX	SCIP	DEBS	Hybrid	Li	SDP
gka1a	50	11.32	0.03	4.55	**0.02**	<1	0.33
gka2a	60	1.60	**0.02**	0.84	**0.02**	<1	0.42
gka3a	70	2817.3	0.93	1335.20	**0.79**	<1	0.93
gka4a	80	88.86	1.12	348.61	**0.86**	<1	1.07
gka5a	50	3.22	0.94	**0.11**	**0.11**	<1	0.29
gka6a	30	0.11	0.37	**0.01**	0.02	<1	0.06
gka7a	30	0.05	0.41	**0.01**	0.02	<1	0.07
gka8a	100	-	**0.08**	634.88	**0.08**	<1	2.11

gkaib problems

Instance	n	CPLEX	SCIP	DEBS	Hybrid	Li	SDP
gka1b	20	0.08	0.29	**0.01**	**0.01**	<1	0.10
gka2b	30	0.1	0.36	**0.01**	**0.01**	<1	2.18
gka3b	40	0.58	0.6	**0.01**	**0.01**	<1	9.44
gka4b	50	1.7	1.15	**0.01**	0.03	<1	18.42
gka5b	60	5.86	1.44	**0.04**	0.07	<1	39.20
gka6b	70	21.75	1.95	**0.15**	0.26	<1	86.56
gka7b	80	59.88	2.52	**0.34**	0.53	<1	212.73
gka8b	90	160.79	3.67	**0.98**	1.43	<1	789.10
gka9b	100	502.79	6.04	**2.66**	3.67	<1	1608.04
gka10b	125	-	9.51	20.7	12.06	**<1**	5104.21

gkaid problems

Instance	n	CPLEX	SCIP	DEBS	Hybrid	Li	SDP
gka1d	100	-	5.90	-	5.84	24	**2.10**
gka2d	100	-	-	-	-	5671	**2.65**
gka3d	100	-	-	-	-	1713	**2.79**
gka4d	100	-	-	-	-	3835	**4.23**
gka5d	100	-	-	-	-	5466	**55.72**
gka6d	100	-	-	-	-	1534	**3.03**
gka7d	100	-	-	-	-	4273	**46.04**
gka8d	100	-	-	-	-	683	**2.94**
gka9d	100	-	-	-	-	2481	**34.08**
gka10d	100	-	-	-	-	1878	**30.97**

on the pure DEBS results on all problem sets though the improvement is marginal for bqp50, gkaib and gkaid.

While our experimental environment is different from that of Li et al., the new hybrid algorithm appears competitive on the Carter type, William type, bqp50, gkaia, and gkaib problems. The hybrid algorithm performs better on the bqp100 problems and worse on the gkaid problems. For the bqp100 problems, the hybrid algorithm outperforms Li et al.'s algorithm by about an order of magnitude.

Compared to the SDP approach, the hybrid algorithm performs basically the same or slightly better for moderate size problems, but significantly less efficiently for larger problems. However for some dense problem instances (gkaib), the hybrid algorithm greatly outperforms the SDP approach. The hybrid algorithm was able to solve all of them quickly, while the SDP approach is in general two orders of magnitudes slower. Overall, the SDP approach still appears to be most efficient for the BQP problems.

3.2.5.5 Discussion

Our results in Section 3.2.4 showed that the DEBS method performs especially well on ILS problems when the residual $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{z}^*\|_2^2$, or equivalently, the noise in the data, is small ($\boldsymbol{z}^*$ is the optimal integer solution). However, a MIP solver performs better than DEBS with small variable domains and large noise. To investigate an analog of the noise in the BQP problem, we use the following linear model: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v}$, where $\boldsymbol{v} \in \mathbb{R}^n$ denotes the noise vector. As mentioned in Section 3.2.4, a large noise vector results a large residual.

In communication applications, a noise vector $\boldsymbol{v}$ of $\sigma * \texttt{randn(m,1)}$, $\sigma \geq 5$ is generally considered as an extremely large noise in the data [11]. We computed the noise vectors in the seven problem sets by substituting the optimal solution or the best solution found within the time limit into the linear model and found that $\sigma > 5$ for all seven problem sets (complete results in Appendix A.2). Therefore these problems can be regarded as problems with very large noise and the poor performance of the DEBS method on the larger problems is consistent with both our previous results and the need to search a larger ellipsoid.

Our results show that combining the DEBS method with a B&C based MIP solver indeed results in better performance when large noise is present in the problem instances. The additional techniques from MIP such as relaxation, cutting planes, and inference appear to complement the presolving, axis-aligned box constraints, and primal heuristic adopted from the DEBS approach.

Analysis of CPLEX solving behaviour shows that the reason that CPLEX is unable to prove optimality for bigger problems (e.g., bqp100 and gkaid) is mainly weakness in the dual bound. Solutions with good quality are found early in the search but the dual bound only improves slowly. For the DEBS method, optimal or near optimal solutions are typically found, though not as quickly as with CPLEX. However, similarly to CPLEX, DEBS is not able to prove optimality since the search space is too large and it does not have an effective dual bounding mechanism. SCIP, however, performs well on the gka1d, bqp50, and bqp100 problems because it is able to make rapid improvements in the dual bound. The lack of such an ability, conversely, appears to explain the very poor performance of SCIP on the Carter and Williams problems. One area for future work is to investigate the performance differences on individual problem sets, in particular, to understand the structure that SCIP appears to be taking advantage of in the problems in which it performs well. The comparison of the hybrid algorithm and the SDP approach shows that the hybrid algorithm seems to outperform the SDP approach on dense problems (e.g., gkaib). Further investigation is required to identify the reasons.

3.3 Combining DEBS with CP

In the previous section, we investigated the combination of DEBS with MIP and demonstrated state-of-the-art and near state-of-the-art results on four problems classes in the communications and OR literature. Here, we investigate another hybridization of DEBS, this time with constraint programming techniques. In particular, we extend DEBS to perform inference on linear constraints, enabling it, for the first time to be applied to problems with linear constraints. We demonstrate our hybrid algorithm on the exact quadratic knapsack problem (EQKP) [99], which has a quadratic objective function, a cardinality constraint, and a knapsack constraint.

3.3.1 Literature Review

Given a set of elements where a distance is specified for each pair of them, the EQKP consists of selecting a subset of elements such that the sum of the distances between the chosen elements is maximized while also satisfying a knapsack constraint. It is an extension of the well studied maximum diversity problem [106], the quadratic knapsack problem [39], and the exact linear knapsack problem [38], which arise in a wide range of real world applications such as wind farm optimization [147, 163] and task allocation [100]. The EQKP consists of one quadratic objective function and two linear constraints, i.e., one knapsack constraint and one cardinality constraint, where all variables are binary. It was first studied by Létocart [99] who proposed a heuristic method for the problem.

For solving EQKPs exactly, the common generic approach in OR is the use of a commercial MIP solver such as CPLEX or Gurobi. These solvers have been extended to reason about quadratic constraints [37] and they are able to outperform several other exact approaches [25]. The other generic approach is the semi-definite programming (SDP) based branch-and-bound algorithm [87], as we saw in the previous section, which is often regarded as the state-of-the-art approach for solving BQPs. EQKP is an extension of the BQP and it can be solved with the SDP approach. However, the SDP approach has not been evaluated on problems with the EQKP structure.

3.3.2 Problem Definition

The EQKP problem is defined as follows:

$$\begin{aligned} \max_{\boldsymbol{x}\in\{0,1\}} \quad & \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} h_{ij}x_i x_j, \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i = K, \\ & \sum_{i=1}^{n} a_i x_i \leq B, \end{aligned} \tag{3.10}$$

where $n \in \mathbb{Z}$, $K \in \mathbb{Z}$, $a_i \in \mathbb{R}^+, \forall i$, $B \in \mathbb{R}^+$, $h_{ij} \in \mathbb{R}, \forall i, j$.

For the simplicity of explanation, we reformulate the EQKP in its minimization form with matrix representation as follows:

$$\min_{\boldsymbol{x}\in\{0,1\}} \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x}, \quad \text{s.t. } \boldsymbol{c}_1^\top \boldsymbol{x} = K,\ \boldsymbol{c}_2^\top \boldsymbol{x} \leq B, \tag{3.11}$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is a symmetric semi-definite positive matrix, $\boldsymbol{f} \in \mathbb{R}^n$ is a vector equal to zeros, $\boldsymbol{c}_1 \in \mathbb{R}^n$ is a vector equal to ones, and $\boldsymbol{c}_2 \in \mathbb{R}^n$ is a vector equal to a_is. Note that $\boldsymbol{H}$ can always be made symmetric semi-definite positive to ensure convexity when all variables are binary [25].

3.3.3 The Hybrid Algorithm

To solve the EQKP with the DEBS method, we need to transform the objective function into its equivalent ILS form through the relationship $\boldsymbol{H} = \boldsymbol{A}^\top \boldsymbol{A}$ and $\boldsymbol{f} = -\boldsymbol{y}^\top \boldsymbol{A}$. Thus, the equivalent ILS problem can be defined as follows:

$$\min_{\boldsymbol{x}\in\{0,1\}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \text{s.t. } \boldsymbol{c}_1^\top \boldsymbol{x} = K,\ \boldsymbol{c}_2^\top \boldsymbol{x} \le B. \tag{3.12}$$

3.3.3.1 Modifying the Reduction

As explained in Section 2.3.4, the reduction phase of DEBS transforms $\boldsymbol{A}$ into an upper triangular matrix $\boldsymbol{R}$ such that the diagonal entries are ordered in non-decreasing order to improve the efficiency of the DEBS search by reducing the branching factor at the top of the search tree [43]. Here we demonstrate how to perform reduction on problems with linear constraints, using the EQKP as an example.

Transforming $\boldsymbol{A}$ to $\boldsymbol{R}$ for the EQKP can be achieved by finding an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and a permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$ such that $\boldsymbol{Q}^\top \boldsymbol{A}\boldsymbol{P} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}$, $\boldsymbol{Q} = \begin{bmatrix}\boldsymbol{Q}_1, \boldsymbol{Q}_2\end{bmatrix}$. Therefore we have:

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \left\|\boldsymbol{Q}_1^\top \boldsymbol{y} - \boldsymbol{R}\boldsymbol{P}^T \boldsymbol{x}\right\|_2^2 + \left\|\boldsymbol{Q}_2^\top \boldsymbol{y}\right\|_2^2.$$

Let $\bar{\boldsymbol{y}} = \boldsymbol{Q}_1^\top \boldsymbol{y}$, $\boldsymbol{z} = \boldsymbol{P}^T \boldsymbol{x}$, and $\bar{\boldsymbol{c}}_2 = \boldsymbol{c}_2 \boldsymbol{P}$, the original EQKP (3.12) is then transformed to the new reduced EQKP:

$$\min_{\boldsymbol{z}\in\{0,1\}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \quad \text{s.t. } \boldsymbol{c}_1^\top \boldsymbol{x} = K,\ \bar{\boldsymbol{c}}_2^\top \boldsymbol{x} \le B. \tag{3.13}$$

Note that applying the permutation matrix $\boldsymbol{P}$ to the original problem changes the order of the variable bounds and the coefficients of the linear constraints. However, since the original lower and upper bounds on the variables are all zeros and ones, and the coefficients of the cardinality constraint are all ones for the EQKP, the new bounds and $\boldsymbol{c}_1$ remain unchanged after applying the permutation matrix $\boldsymbol{P}$. For problems with general variable bounds, i.e., $\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}$, the reordering can be done with $\bar{\boldsymbol{l}} = \boldsymbol{P}^\top \boldsymbol{l}$ and $\bar{\boldsymbol{u}} = \boldsymbol{P}^\top \boldsymbol{u}$.

After the optimal solution $\boldsymbol{z}^*$ to the new EQKP problem (3.13) is found, the optimal solution, $\boldsymbol{x}^*$, to the original EQKP problem (3.12) can be recovered with the relationship $\boldsymbol{x}^* = \boldsymbol{P}\boldsymbol{z}^*$.

3.3.3.2 Modifying the Search

From a constraint programming perspective, DEBS does three things. First, the search strategy implies a fixed variable ordering heuristic $(z_n, z_{n-1}, \ldots, z_k, \ldots, z_1)$ that is determined after the reduction phase, where the variables are reordered with the permutation matrix as $\boldsymbol{z} = \boldsymbol{P}^\top \boldsymbol{x}$. Second, through the calculation of c_k, i.e., the center of the ellipsoid in the k-th dimension (see Section 2.3.4), DEBS provides

a value ordering heuristic: for z_k, integer values closer to c_k are preferred. Third, the inequalities described in Section 2.3.4.1 induce an interval domain for each z_k. DEBS is essentially the enumeration of these domains under the prescribed variable and value orderings. We use this insight to integrate linear constraints into DEBS in a straightforward way: linear constraints are made arc consistent (AC) to further reduce the domains of the z_k variables.

A linear constraint is defined as $\underline{b} \leq \boldsymbol{a}^\top \boldsymbol{z} \leq \bar{b}$, where $\underline{b}$, $\bar{b} \in \mathbb{R}^n \cup \{\pm\infty\}$, and $\boldsymbol{a} \in \mathbb{R}^n$ is the coefficients of the constraint. Given a linear constraint, let

$$\underline{\alpha}_k = \min\{\boldsymbol{a}^\top \boldsymbol{z} - a_k z_k \mid \boldsymbol{l} \leq \boldsymbol{z} \leq \boldsymbol{u}\} \quad \text{and} \quad \bar{\alpha}_k = \max\{\boldsymbol{a}^\top \boldsymbol{z} - a_k z_k \mid \boldsymbol{l} \leq \boldsymbol{z} \leq \boldsymbol{u}\}$$

be the minimal and maximal values that $\boldsymbol{a}^\top \boldsymbol{z}$ can achieve over all variables except z_k with respect to the variable bounds. These values can be computed by substituting the z_k variables with their lower or upper bounds, depending on the signs of the a_k variables. The propagation rule for each integer variable z_k is then defined as follows:

$$\left\lceil \frac{\underline{b} - \bar{\alpha}_k}{a_k} \right\rceil \leq z_k \leq \left\lfloor \frac{\bar{b} - \underline{\alpha}_k}{a_k} \right\rfloor \quad \text{if } a_k > 0, \tag{3.14}$$

$$\left\lceil \frac{\bar{b} - \underline{\alpha}_k}{a_k} \right\rceil \leq z_k \leq \left\lfloor \frac{\underline{b} - \bar{\alpha}_k}{a_k} \right\rfloor \quad \text{if } a_k < 0. \tag{3.15}$$

Although the idea of propagating the linear constraint is straightforward and not novel, the implementation involves maintaining useful information efficiently as the search moves from assigning z_{k+1} to z_k. Recall that variable z_k is instantiated at level k (see Section 2.3.4) in the DEBS search. As a variable is fixed when the search moves down the levels, the values $\underline{\alpha}_k$ and $\bar{\alpha}_k$ actually consist of two parts: the sum of all the variables that are fixed already (z_{k+1} to z_n), and the sum of the maximum or minimum values that the unfixed variables (z_1 to z_{k-1}) can achieve, according to the variable bounds. Below we show how to update $\underline{\alpha}_k$ and $\bar{\alpha}_k$ using the cardinality constraint as an example.

For the cardinality constraint, we have $a_k = 1, \forall k$, and $\underline{b} = \bar{b} = K$. Therefore, using Equation (3.14), the lower bound L_k and upper bound U_k imposed by the cardinality constraint can be derived as follows:

$$L_k = K - \sum_{i=k+1}^{n} z_i - \sum_{i=1}^{k-1} u_i \quad \text{and} \quad U_k = K - \sum_{i=k+1}^{n} z_i - \sum_{i=1}^{k-1} l_i,$$

where $\sum_{i=1}^{k-1} l_i = 0$ and $\sum_{i=1}^{k-1} u_i = k-1$, since z_is are binary for the EQKP.

During the search, the sum of the fixed values $S = \sum_{i=k+1}^{n} z_i$ is only changed when moving up or down between two adjacent levels. Therefore, we can use a single variable to keep track this value, efficiently updating S rather than computing from scratch. When the search backtracks from level k to $k+1$, we set $S = S - z_k$ to reverse the assignment of z_k. Similarly, when the search moves down from level k to $k-1$, we set $S = S + z_k$ to take into account the current assignment of z_k.

Equations (3.14) and (3.15) are general and they can be applied to all types of linear constraints. Similarly, each linear constraint keep tracks of its own sum of the current assigned values S. Therefore our extended DEBS can be used to solve problems with any number of linear constraints of any type. The variable domain at each level is the intersection of all the bounds imposed by the linear constraints.

In addition, some types of constraints, e.g., the cardinality constraint as shown above and the knap-

sack constraint, can be specialized from the general linear constraint formulation and be propagated more efficiently. For the EQKP, a variable's domain does not become empty unless the cardinality number or the capacity is exceeded, which only happens when the variable instantiation violates the cardinality constraint or the knapsack constraint at the current level. Therefore, we can do a *lazy* version of AC that only filters the variable at the current level. The resulting filtering thus runs faster than maintaining AC directly, while exploring the exact same nodes as AC.

A naive extension of the DEBS method for solving problems with linear constraints can be achieved by using the linear constraints only as "checkers": the constraints simply return true or false when all the variables in their scope are instantiated. If all linear constraints agree on an instantiation, it is accepted as an integer solution and the search proceeds correspondingly.

Solving Non-Binary Problems. Our extended DEBS method can solve non-binary problems without any modification. However, it requires that the quadratic objective function be inherently convex, as convexifying a non-binary problem is not possible in general. The reduction and the propagation on the linear constraints are not affected by the variable domains and therefore remain the same.

3.3.4 Experimental Results: EQKP Problems

3.3.4.1 Setup

The CPU time limit for each run on each problem instance is 3600 seconds. All experiments were performed on a Intel(R) Xeon(R) CPU E5-1650 v2 3.50GHz machine (in 64 bit mode) with 16GB memory running MAC OS X 10.9.2 with one thread. The hybrid approach is written in C. We use CPLEX Optimization Studio v12.6 and the SDP solver BiqCrunch downloaded from the website [88] for comparison. Both solvers are executed with their default settings. For the SDP solver, there are four specialized versions that deal with problem-specific structures. Three of them are consistent with the EQKP problem and the SDP results presented are the best of the three versions for each individual problem instance, representing the "virtual best" SDP solver. We report the arithmetic mean CPU time "arith", and the shifted geometric mean CPU time "geo" to find and prove optimality for each problem set.

3.3.4.2 Test sets

We use five sets of the benchmark instances with different sizes generated in the same way as Létocart et al. [99], with 10 instances in each set. For additional comparison, we relax the binary domains $x_j \in \{0,1\}$ to $x_j \in \{0,1,2\}$, meaning that we have the option of selecting two "copies" of each element when maximizing the quadratic objective function but $h_{jj} = 0$ (see Formulation 3.10). To the best of our knowledge, this problem has not been studied in the literature.

In order to ensure convexity, we compute the smallest eigenvalue for the $\boldsymbol{H}$ matrix of each problem and let it be λ_{min}. If λ_{min} is negative, i.e., the problem is non-convex, we apply the perturbation vector $\boldsymbol{u} = (-\lambda_{min} + 0.001)\boldsymbol{e}$ such that the original objective function is transformed to: $\min_{\boldsymbol{x}\in\{0,1\}} \frac{1}{2}\boldsymbol{x}^\top \bar{\boldsymbol{H}}\boldsymbol{x} + \bar{\boldsymbol{f}}^\top \boldsymbol{x}$, where $\bar{\boldsymbol{H}} = \boldsymbol{H} + \mathrm{diag}(\boldsymbol{u})$ and $\bar{\boldsymbol{f}} = \left(\boldsymbol{f} - \frac{1}{2}\boldsymbol{u}\right)^\top$. Note that this convex formulation has the same optimal solution as the original one. For the DEBS method, we use Cholesky decomposition on the perturbed matrix $\bar{\boldsymbol{H}}$ to obtain matrix $\boldsymbol{A}$ in the ILS formulation (3.12), and we obtain $\boldsymbol{y}$ from the relationship

$\bar{\boldsymbol{f}} = -\boldsymbol{y}^\top \boldsymbol{A}$, which gives us $\boldsymbol{y} = -(\bar{\boldsymbol{f}}\boldsymbol{A}^{-1})^\top$. If the problem is originally convex, we obtain $\boldsymbol{A}$ and $\boldsymbol{y}$ with $\boldsymbol{H}$ and $\boldsymbol{f}$.

3.3.4.3 Results of the EQKP

We first compare the three different implementation of the hybrid DEBS+CP algorithms: DEBS+arc consistency (AC), DEBS+lazy arc consistency (LAC), and DEBS+checking (CHECK). From Table 3.3, we can see that DEBS+LAC performs the best, while DEBS+CHECK is, unsurprisingly, orders of magnitude slower than DEBS+AC and DEBS+LAC. This result suggests that the propagation is critical to the performance.

Second, we compare the best hybrid algorithm to the other solvers. From Table 3.4, it is clear that the DEBS+LAC hybrid algorithm performs the best both for the EQKP and its relaxed problem. For the EQKP, the hybrid algorithm is on average one to two orders of magnitude faster than CPLEX and it strictly dominates CPLEX on each problem instance in our experiments. The SDP approach performs the worst among the three approaches. These results are surprising given the strong performance for solving the BQPs, and the fact that these results are the best per instance results over the three BiqCrunch variations. Analysis of CPLEX and the SDP solving behaviour shows that the reason that both approaches are unable to prove optimality for larger problems is mainly the weakness in the dual bound.

	{0,1} Problems					
n	DEBS+AC		DEBS+LAC		DEBS+CHECK	
	arith	geo	arith	geo	arith	geo
10	**0.01**	0.01	**0.01**	0.01	**0.01**	0.01
20	0.02	0.02	**0.01**	0.01	1.34	1.34
30	4.16	3.39	**0.72**	0.68	1434.52	1434.51
40	709.34^1	153.03^1	**115.64**	31.91	-	-
50	1005.54^2	300.26^2	**317.83**	49.87	-	-

Table 3.3: A comparison of DEBS+AC, DEBS+LAC and the DEBS+CHECK approach for the EQKP ({0,1} Problems). Bold numbers indicate the best approach for a given problem set. The superscripts indicate the number of instances not solved to optimality within 3600 seconds. The symbol '-' means that none of the 10 problem instances were solved to optimality within 3600 seconds.

Interestingly, the running times for all three approaches increase significantly when K is increased. The reason is that, during the search, when the sum of the already fixed variables at a node is equal to K, we know that the unfixed variables have to be set to zero to satisfy the cardinality constraint. Intuitively, if K is small, such solutions are early in the search tree, as fewer branching decisions are required to reach such nodes. For the same reason, we expect that the running time to be also decreased when K is further increased towards n.

From Table 3.4, it is observed that while the relaxed domain makes the problem much more difficult to solve, the hybrid algorithm still dominates CPLEX. The SDP approach cannot be used to solve the relaxed problem without transforming the problem back to binary domains at the cost of introducing additional variables. Therefore, we perform this transformation using the standard technique by representing the integer variables with their binary expansion [55], i.e., we replace each $x_j \in \{0, 1, 2\}$ with $y_{j0} + 2y_{j1}$, where y_{j0} and y_{j1} are binary. Results show that the SDP approach does not solve the relaxed

{0,1} Problems

n	DEBS+LAC		DEBS+LAC+Red.		CPLEX		SDP	
	arith	geo	arith	geo	arith	geo	arith	geo
10	**0.01**	0.01	**0.01**	0.01	0.04	0.04	0.14	0.14
20	**0.01**	0.01	**0.01**	0.01	0.18	0.18	4.94	4.35
30	0.72	0.68	**0.69**	0.65	18.82	12.21	336.94	152.53
40	115.64	31.91	**109.63**	30.78	1274.05^{2}	401.5^{2}	2374.78^{4}	1238.91^{4}
50	**317.83**	49.87	329.31	50.91	1810.34^{6}	1006.41^{6}	3237.45^{7}	3119.11^{7}

{0,1,2} Problems

n	DEBS+LAC		DEBS+LAC+Red.		CPLEX		SDP	
	arith	geo	arith	geo	arith	geo	arith	geo
10	**0.01**	0.01	**0.01**	0.01	**0.01**	0.01	3567.75^{9}	3566.43^{9}
20	**0.02**	0.02	**0.02**	0.02	0.26	0.25	-	-
30	1.05	0.95	**0.99**	0.90	11.49	6.78	-	-
40	**276.28**	65.86	308.52	69.78	1203.59^{2}	281.98^{2}	-	-
50	$\mathbf{2354.50}^{6}$	1216.43^{6}	2406.48^{6}	1262.99^{6}	3149.80^{8}	2700.36^{8}	-	-

Table 3.4: A comparison of CPLEX, DEBS+LAC and the SDP approach for the EQKP ({0,1} Problems) and its problem variation ({0,1,2} Problems). Bold numbers indicate the best approach for a given problem set. The superscripts indicate the number of instances not solved to optimality within 3600 seconds. The symbol '-' means that none of the 10 problem instances were solved to optimality within 3600 seconds.

problem efficiently due to the very slow improvement of the dual bound.

It is interesting to observe that the reduction does not seem to improve the performance for the EQKP as it does on the BQP problems without general linear constraints [89]. Reduction even worsens performance on the relaxed problems. It may, therefore, be of interest to develop new reduction algorithms that achieve improvements similar to the original reduction but in the presence of linear constraints.

For the sake of completeness, we also hybridize our DEBS+LAC approach with MIP as done in Section 3.2 to solve the EQKP. However, the results are worse than that of DEBS+LAC. For example, the average running time of the EQKP ({0,1} Problems) are 0.01, 0.01, 0.69, 208.02 and 1170.52 seconds, for the five problems. In addition, for instances of size 50, 1 out of 10 instances could not be solved to optimality within the time limit. The reason is because that the MIP component does not contribute to the overall solving capability, as the dual bound is very weak.

3.4 Conclusion

In this chapter, we propose two hybrid algorithms that combine techniques from DEBS and MIP and CP, respectively.

The hybridization of DEBS and MIP implements DEBS-based presolving, axis-aligned box constraints, and a primal heuristic in a MIP solver to solve the three types of ILS problems that arise in signal processing applications and the classical BQP problem in the OR field.

For the ILS problem, we show that the hybrid algorithm outperforms both the MIP and DEBS approaches individually, while under-performing MIP, only for box-constrained problems with large

noise. Our results demonstrate that although DEBS is often considered the state-of-the-art approach in the communications literature for solving ILS problems, it is possible to substantially improve upon its performance by hybridizing it with complementary technology from another field of study. The hybrid algorithm can be especially useful when the ILS problems have to be solved a large number of times or when the running time is very limited as in some important applications [146].

We also conduct the first empirical study comparing the performance of a MIP solver, the DEBS method, and the hybrid algorithm for three types of the ILS problems. We performed extensive experiments on problems with various attributes and identified key characteristics impacting the performance of each approach: noise, the size of the constraining box, and the size of the problem. Our experimental results demonstrate that the DEBS method is both significantly better and significantly worse than the MIP solver depending on the problem characteristics. DEBS is more sensitive to the noise while MIP is more sensitive to the size of the variable domains. The DEBS method performs better than MIP when noise is small, regardless of the size of the problem or the size of the box, while MIP is two orders of magnitude better for all the box-constrained problems and when the noise is large. This latter result is notable as it shows that for some problem characteristics, an off-the-shelf solver out-performs the specialized technique typically used in the communications literature. However, in the final analysis it is the combination of the specialized approach and the MIP solver in the form of the proposed hybrid algorithm that results in the best problem-solving performance.

For the BQP problems, the hybridization of DEBS and B&C out-performs both the B&C and DEBS based approaches across all problem sets. The improvement is substantial for some problem sets, though only marginal in others. Compared to the best special-purpose algorithms, Li et al.'s branch-and-bound algorithm [102] and an SDP approach [87], the new hybrid algorithm is competitive though overall not as strong as the SDP algorithm. We, therefore, conclude that the hybridization of DEBS with B&C is a strong contender on BQP problems, though the SDP approach remains the state of the art.

Our results also show that the DEBS performs much better than CPLEX and about even with SCIP, though different problem sets show markedly different relative performance. The DEBS algorithm from the communications literature, therefore, is at least competitive with state-of-the-art B&C MIP approaches to binary quadratic problems.

For the hybridization of DEBS and CP, the hybrid algorithm is implemented as an extension to DEBS. The core of our extension required the modification of the DEBS approach to incorporate linear constraints. We did this by adopting standard linear constraint propagation algorithms. Results on the exact quadratic knapsack problem (EQKP) and a variation showed that our algorithm outperforms both the state-of-the-art MIP and SDP approaches.

Our work on the hybrid algorithms has inspired us to explore other possibilities of using geometric information from DEBS to enhance MIP and CP solving. Specifically, in Chapter 5 we develop inference rules and search strategies to solve the strictly convex quadratically-constrained problems.

In the next chapter, we focus on an important case of the IQCP: the variance minimization problem. Such problem arise in many applications such as load balancing scheduling problems. We again develop a hybrid algorithm that combines techniques from MIP and CP to solve two real world problems, i.e., the load balancing nurse scheduling problem and the balanced academic curriculum problem.

Chapter 4

Constraint Integer Programming for Variance Minimization: An Application to Load Balancing Problems

In Chapter 3 we showed how MIP, CP and DEBS can be hybridized to solve some special cases of the IQCP efficiently. The hybridization of MIP and DEBS can be regarded as a component enhancement, since different aspects of DEBS are incorporated into MIP as a presolving technique, cutting planes, and a primal heuristic. The hybridization of CP and DEBS is done in a different way and it can be seen as a node processing enhancement, i.e., the propagation from the linear constraints is used to prune variable domains in the DEBS search.

In this chapter, we study a even tighter integration, this time combining MIP, CP and DEBS together. The integration is done in the paradigm of Constraint Integer Programming (CIP) (see Section 2.3.3 and the references therein) which integrates CP and MIP modelling and solving techniques. Specifically, at each node, we perform MIP routines such as LP relaxation solving and CP routines such as constraint propagation. In addition, we strengthen the problem formulation with cutting planes based on a CP perspective on problem structure and the geometrical reasoning from DEBS. We demonstrate that the CIP approach can be used to efficiently solve the variance minimization problem, an important class of IQCPs. In this chapter, we focus our study on load balancing problems, which appear frequently in scheduling applications.

We note that although our CIP approach was clearly the state-of-the-art in our previous conference publication [93], substantial improvements have been made on the MIP and CP solvers used for comparison. Therefore we have rerun all the experiments with the latest versions of the solvers.

4.1 Introduction

Load balancing problems are concerned with the assignment of a given workload to different resources such that the loads are as evenly distributed as possible. Common objectives include minimizing the

maximum load [41, 78], the gap between the maximum and minimum loads [116], the sum of deviations from the mean load (L_1-norm), the sum of squares deviations from the mean load (L_2-norm), and the maximum deviation from the mean load (L_∞-norm) [115, 136]. Among these different load balancing criteria, it has been shown that minimizing the L_2-norm achieves the best performance in terms of evenly distributing the loads across all resources [45, 138, 139], as the L_2-norm is more sensitive to extreme values. Indeed, in real world load balancing problems, e.g., workforce scheduling where fairness among workers is crucial, many small deviations from the mean are often considered more desirable than a few large deviations [139]. Therefore, we are interested in the L_2-norm, i.e., minimizing the variance of the loads.

In this chapter, we study two real world applications, the load balancing nurse-to-patient assignment problem (NPAP) and the balanced academic curriculum problem (BACP). The NPAP assigns nurses to a hospital zone (e.g., a nursery room) and to patients within the zone to minimize the variance in the total acuity of patients assigned to each nurse. The BACP is concerned with assigning courses to academic periods and balancing the course load within each period, subject to some constraints on course prerequisites.

Constraint programming based approaches, combining the SPREAD constraint [124] with problem-specific search heuristics, are currently the state-of-the-art for solving the NPAP [121, 122] and the BACP [115, 121] exactly. In this chapter, we address the problem with two approaches that, to our knowledge, have not yet been applied: mixed integer quadratic programming (MIQP) and CIP [3, 4].

Mixed integer linear programming (MILP) techniques have been extended to reason about convex quadratic constraints [37], allowing MIQPs to be solved by commercial MILP solvers. Since variance minimization can be modelled as an MIQP and underlying MILP technology has seen substantial improvement over the past few years [86], we develop and apply an MIQP model to the problem. The model is natural, given the direct representation of the quadratic constraints, however we show that model performs worse using CPLEX than the best known CP model implemented in OSCAR for the NPAP.[1]

We then improve the MIQP model in two orthogonal directions. First, in the CIP framework, we exploit the idea of *augmented global constraints* that embed both inference techniques in the form of bounds propagation and relaxation-based reasoning via constraint-specific linear relaxations and cutting planes. These techniques are functional extensions to global constraints (beyond filtering) that can be implemented in any solver that allows linear relaxations and the dynamic addition of cutting planes. In particular, we augment the MIQP model to include the QUADRATIC [23], GCC, and SPREAD global constraints. For the first two constraints, the propagation, relaxation, and cutting plane techniques are taken from the literature. For the SPREAD constraint, we implement the bound consistency algorithm due to Schaus et al. [137] and develop novel relaxation and cutting plane techniques. Our augmented global constraints can be soundly used in any models in which the unaugmented global constraints appear. The second direction of improvements is to modify the branching heuristics. Given the structure of the NPAP and the BACP, we develop two simple, static variable ordering heuristics that influence the default, general branching rules of the CIP solver, SCIP [3, 5].

For the NPAP, we first note that while our CIP approach developed in 2014 [93] underperforms a recent decomposition method [122], it is still of interest from the perspective of the "model and

[1] In our previous publication [93], we used COMET as the CP solver. We switched to OSCAR as COMET is no longer supported by the developers.

solve" paradigm and the goal of declarative problem solving [60]. Problem decomposition requires more knowledge and work by the modeler and restricts the extensibility of the model, for example, to the addition of inter-zone constraints. Therefore in the rest of the chapter, we focus our study on models without decomposition. Our empirical results show that the CIP models without problem-specific branching heuristics perform slightly worse than the MIQP model in CPLEX. However, the existing CP model still performs the best. When problem-specific branching priorities are used, the best CIP model outperforms both the MIQP model in CPLEX and the previous state-of-the-art CP model. The best CIP model includes all three global constraints and results in at least an order of magnitude improvement over MIQP. Compared to the default heuristics, the problem-specific heuristics find and prove optimal solutions with 30 to 40 times less effort in terms of both search tree size and run-time. Subsequent analysis shows that the improved performance arises due to the rapid improvement in both the primal (upper) bound and the dual (lower) bound early in the search.

For the BACP, both our new MIQP model and CIP models with problem-specific heuristic outperform the previous best known CP model. In addition, the relative performance of the CIP models follow the same trend as those in the NPAP, reinforcing our results on the effectiveness of the augmented global constraints.

4.1.1 Contributions

The contributions of this chapter are as follows:

- Within the framework of the augmented global constraint, we develop relaxations and cutting planes for the spread constraints and implement them in the CIP solver SCIP. We show how various global constraints cannot only embed their traditional filtering algorithm but also linear relaxations strengthened with cutting planes. In terms of reusability, the augmented global constraints can be applied to any problems that contain e.g., spread constraints, in any solver that allows linear relaxations and the dynamic addition of cutting planes.

- We study the structure of the load balancing problems and propose problem-specific branching rules that guide the search in an efficient way. Our CIP models with branching rules outperform the previous best known non-decomposition based CP model for the NPAP and the best known CP model for the BACP.

- We propose novel MIQP models that solve the load balancing problems in the CP literature and perform the first empirical study on the NPAP and the BACP that analyze the performance of MIP and CP. Results show that the MIQP model achieves the state-of-the-art for the BACP.

A preliminary report on the work in this chapter has previous appeared in a conference publication [93].

4.1.2 Organization

This chapter is organized as follows. We present the two load balancing problems in Sections 4.2 and 4.3. In each section, we first review relevant literature and present the problem definition. We then introduce the base CP model in the literature and the new MIQP and CIP models. We address our primary contribution, the CIP components, in Section 4.4, which presents the augmented global constraints used

in the CIP models. We discuss the branching heuristics in Section 4.5. Sections 4.6 and 4.7 provide computational results and discussions for the two load balancing problems, respectively. We conclude in Section 4.8.

4.2 The Load Balancing Nurse-to-Patient Assignment Problem (NPAP)

4.2.1 Background

The load balancing nurse-to-patient assignment problem (NPAP) requires the assignment of nurses to new-born infant patients across different zones in a hospital [116]. Each infant is hospitalized in one of the rooms, or zones, in the nursery and requires a certain amount of care depending on the acuity of his/her condition. The problem has two main decisions. First, each nurse has to be assigned to a zone in which he/she will work for the entire shift. Second, each nurse has to be assigned to a set of patients in the same zone. The objective is to minimize the variance of the total acuity assigned to each nurse. Such an assignment will avoid overloading the nurses, which can result in stress and poor quality of the care.

The problem was originally modeled and solved as an MILP with a linear objective function that minimizes the sum of the differences between the maximum and minimum assigned workload in each zone [116]. However, although the objective function ensures the workload of each zone is evenly distributed, the workload difference between nurses in different zones may be large.

Schaus et al. [138, 139] proposed a CP model that directly minimizes the standard deviation of the nurses' workloads using the bounds consistency SPREAD constraint [124]. Results showed significant improvements in both solution quality and computational efficiency compared to the MILP model. The CP approach is able to solve problems with two zones exactly, but not very efficiently without using an approximation of the number of nurses assigned to each zone and further decomposition. While the approximation and decomposition techniques solve the problem quickly, optimality is not guaranteed. More recently, Pesant [121] solved the single-zone problem with domain consistency SPREAD constraint and proposed an exact CP-based decomposition method for the multi-zone problem [122].

4.2.2 Mathematical Models

4.2.2.1 Problem Definition

The load balancing NPAP is defined by a finite set of m patients, a finite set of n nurses, and a finite set of p zones. In addition, let P_k denote the set of patients in zone k, thus $\{P_1, \ldots, P_p\}$ forms a partition of $\{1, \ldots, m\}$. For each patient, i, a non-negative integer, A_i, represents his/her acuity. The mean of the nurses' workload is therefore computed as $\mu = \sum_{i=1}^{m} A_i/n$. The sum of the acuity levels of the patients assigned to a nurse cannot exceed $MaxAcuity$ and the total number of assigned patients cannot exceed $MaxPatients$. Each patient can only be assigned to one nurse, and each nurse can only be assigned to one zone. The objective is to minimize the variance or, equivalently, the standard deviation of the nurses' workload, measured as the total acuity assigned to the nurses.

$$\min \quad \sigma \tag{4.1}$$

$$\text{s.t.} \quad \texttt{spread}(\{W_1, \ldots, W_n\}, \mu, \sigma), \tag{4.2}$$

$$\texttt{multiknapsack}(\{N_1, \ldots, N_m\}, \{A_1, ..., A_m\}, \{W_1, \ldots, W_n\}), \tag{4.3}$$

$$\texttt{cardinality}(\{N_1, \ldots, N_m\}, \{1, \ldots, n\}, \{1, \ldots, MaxPatients\}), \tag{4.4}$$

$$\texttt{pairwiseDisjoint}(\{Z_1, \ldots, Z_p\}), \tag{4.5}$$

$$Z_k = \bigcup_{i \in P_k} N_i, \qquad k = 1, \ldots, p \tag{4.6}$$

$$W_j \in \{min\{A_i\}, \ldots, MaxAcuity\}, \qquad j = 1, \ldots, n \tag{4.7}$$

$$N_i \in \{1, \ldots, n\}. \qquad i = 1, \ldots, m \tag{4.8}$$

Figure 4.1: The CP model of Schaus et al. [139] for the NPAP.

4.2.2.2 The CP Model

In the state-of-the-art exact CP model [139], the decision variables are defined as follows:

- N_i denotes the nurse assigned to patient i.
- W_j denotes nurse j's workload.
- σ denotes the standard deviation of the variables $W_1, \ldots, W_n$.

The CP model is shown in Fig. 4.1. Constraint (4.2) is the SPREAD constraint, which relates a set of variables to their mean and standard deviation. Constraint (4.3) is a global packing constraint that governs the packing of items, corresponding to patients with "size" equal to their acuity levels, into bins corresponding to the workload of each nurse. Constraint (4.4) is the GCC placing the limit on each nurse in terms of number of assigned patients. Constraint (4.5) expresses that a nurse can only work in one zone during the shift, where Z_k is the set of nurses assigned to zone k, i.e., $Z_k = \bigcup_{i \in P_k} N_i$. The `pairwiseDisjoint` constraint enforces pairwise empty intersections among variables representing the set of nurses working in each zone. Constraint (4.7) expresses bounds on the workload of each nurse. Since there are always more patients than nurses, each nurse will be assigned to at least one patient and, therefore, the W_j variables have a lower bound equal to the minimum acuity among all patients.

A customized search heuristic is an important aspect of the success of the CP model. First, the symmetry arising from the interchangeability of the nurses is dynamically broken during search by exploiting the equivalence among all nurses who have not yet been assigned a patient. Second, problem-specific variable and value ordering heuristics are implemented: the unassigned patient with the highest acuity is selected and assigned to the nurse with the current smallest workload.

4.2.2.3 The New MIQP Model

We propose a new mixed integer quadratic programming (MIQP) model for the problem that is mathematically equivalent to the CP model. In addition to μ, σ, and the W_j variables, we define two additional decision variables as follows:

- x_{ij} is equal to 1 if patient i is assigned to nurse j.
- z_{jk} is equal to 1 if nurse j is assigned to work in zone k.

$$\min \quad \sigma \tag{4.9}$$

$$\text{s.t.} \quad \sigma \geq \sqrt{\frac{\sum_{j=1}^{n}(W_j - \mu)^2}{n}}, \tag{4.10}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, m \tag{4.11}$$

$$\sum_{k=1}^{p} z_{jk} = 1, \qquad j = 1, \ldots, n \tag{4.12}$$

$$W_j = \sum_{i=1}^{m} x_{ij} A_i, \qquad j = 1, \ldots, n \tag{4.13}$$

$$\sum_{i \in P_k} x_{ij} \leq z_{jk} MaxPatients, \qquad j = 1, \ldots, n, \quad k = 1, \ldots, p \tag{4.14}$$

$$W_j \leq W_{j+1}, \qquad j = 1, \ldots, n-1 \tag{4.15}$$

$$x_{ij} \in \{0, 1\}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{4.16}$$

$$z_{jk} \in \{0, 1\}, \qquad j = 1, \ldots, n, \quad k = 1, \ldots, p \tag{4.17}$$

$$W_j \in [min\{A_i\}, MaxAcuity]. \qquad j = 1, \ldots, n \tag{4.18}$$

Figure 4.2: The MIQP model for the NPAP.

The MIQP model is given in Fig. 4.2. The objective function is stated in (4.9). Constraint (4.10) specifies the quadratic relationship between the standard deviation, σ, and the workload variables, W_j. Constraint (4.11) ensures that each patient is assigned to exactly one nurse. Constraint (4.12) ensures that each nurse is assigned to exactly one zone. Constraint (4.13) calculates the workload of each nurse by summing over all patients. Constraint (4.14) ensures that the maximum number of patients assigned per nurse does not exceed the capacity of the nurse and that a nurse is only assigned patients from his/her assigned zone. Constraint (4.15) is the symmetry breaking constraint.

4.2.2.4 The New CIP Model

We propose a novel CIP model that adds global constraints to the MIQP model. To do this, we include the N_i variable with the same meaning as in the CP model: the index of the nurse assigned to patient i. We link N_i to x_{ij} as follows:

$$N_i = \sum_{j=1}^{n} x_{ij} j. \qquad i = 1, \ldots, m \tag{4.19}$$

The global constraints added to the MIQP model to form the CIP model are Constraints (4.2) and (4.4) from the CP model (Fig. 4.1). The complete CIP model is the MIQP model in Fig. 4.2 plus Constraints (4.2), (4.4) and (4.19), .

Note that there is no need for an explicit QUADRATIC global constraint in the CIP model. The SCIP solver used for the CIP models recognizes the quadratic nature of Constraint (4.10) and automatically includes the QUADRATIC constraint.

4.3 The Balanced Academic Curriculum Problem (BACP)

4.3.1 Background

The balanced academic curriculum problem (BACP) is concerned with scheduling courses in different teaching periods at universities. Each course has an academic workload and may have a set of prerequisite courses, which define the partial order of the courses that a student must follow. The goal of the BACP is to assign courses to periods where the academic loads of the courses are balanced within each period, while satisfying the prerequisite requirement. In addition, the number of courses assigned to each period must satisfy given upper and lower bounds.

The BACP was first introduced by Castro and Manzano [41] and solved with CP, where a linear objective function is used to minimize the maximum academic load of a period. The problem was then tackled by Hnich et al. [77, 78] with an hybrid CP/MIP approach and Lambert et al. [95] with another hybrid approach that combines CP and genetic algorithms. Both hybrid approaches showed substantial improvement over the pure CP approaches. However, all of the work above considered only the linear objective function.

Monette et al. [115] introduced different balancing criteria to the BACP, including the variance minimization of the academic loads. Schaus [136] and Monette et al. [114] solved the variance minimization version of the BACP with the SPREAD constraint that provides specific inference based on the relationship between the standard deviation and the mean. However, in both works, a simplified version of the BACP is studied, where the problem does not consider the cardinality constraint that limits the number of courses per period. More recently, Pesant [121] solved the BACP with the domain consistency SPREAD constraint, where both the original and the simplified problems are studied. Results show that the simplified problem can be solved to optimality much faster than the original problem with CP, either using the bounds consistency algorithm or the domain consistency algorithm. The CP approach with the domain consistency SPREAD constraint represents the current state-of-the-art for both the simplified and original problems.

4.3.2 Mathematical Models

4.3.2.1 Problem Definition

The BACP is defined by a finite set of n periods and a finite set of m courses. For each course, i, a non-negative integer, L_i, represents the load of the course. In addition, a finite set of $Prerequisites$ defines the pairwise relationship $(i, \bar{i})$ such that course i must be assigned to a period prior to course $\bar{i}$, $\forall (i, \bar{i}) \in Prerequisites$. The mean of the loads is computed as $\mu = \sum_{i=1}^{m} L_i/m$. The number of the courses assigned to a period is restricted to be between $MinCourses$ and $MaxCourses$. Each course can only be assigned to one period. The objective is to minimize the variance of the loads assigned to the periods.

4.3.2.2 The CP Model

The decision variables are defined as follows:

- N_i denotes the period assigned to course i.
- W_j denotes the academic load of period j.

$$\min \quad \sigma \tag{4.20}$$

$$\text{s.t.} \quad \texttt{spread}(\{W_1, \ldots, W_n\}, \mu, \sigma), \tag{4.21}$$

$$\texttt{multiknapsack}(\{N_1, \ldots, N_m\}, \{L_1, ..., L_m\}, \{W_1, \ldots, W_n\}), \tag{4.22}$$

$$\texttt{cardinality}(\{N_1, \ldots, N_m\}, \{1, \ldots, n\}, \{MinCourses, \ldots, MaxCourses\}), \tag{4.23}$$

$$N_i < N_{\bar{i}}, \qquad \forall (i, \bar{i}) \in Prerequisites \tag{4.24}$$

$$W_j \in \{min\{L_i\}, \ldots, \sum_{i=1}^{m} L_i\}, \qquad j = 1, \ldots, n \tag{4.25}$$

$$N_i \in \{1, \ldots, n\}. \qquad i = 1, \ldots, m \tag{4.26}$$

Figure 4.3: The CP model of Schaus [136] for the BACP.

- σ denotes the standard deviation of the variables $W_1, \ldots, W_n$.

The CP model is shown in Fig. 4.3. Similar to the CP model of NPAP, Constraint (4.21) is the SPREAD constraint, which relates the load variables to their mean and standard deviation. Constraint (4.22) is the global packing constraint and Constraint (4.23) is the GCC that restricts the number of assigned courses to each period. Constraint (4.24) enforces the precedence relations defined by the prerequisite requirement. Finally, Constraint (4.25) places bounds on the loads of each period.

4.3.2.3 The New MIQP Model

We propose a new MIQP model to solve the BACP. In addition to μ, σ, and the W_j variables, we define the additional decision variable as follows:

- x_{ij} is equal to 1 if course i is assigned to period j.

The MIQP model is given in Fig. 4.4. The objective function (4.27) minimizes the standard deviation. Constraint (4.28) specifies the relationship between the standard deviation, σ, and the academic load variables, W_j. Constraint (4.29) ensures that each course is assigned to exactly one period. Constraint (4.30) calculates the academic load of each period by summing over all courses. Constraint (4.31) enforces the minimum and the maximum number of courses assigned to each period. Constraint (4.32) is the prerequisite constraint.

4.3.2.4 The New CIP Model

We propose a novel CIP model and add the global constraints to the MIQP model in the same way that we did for the NPAP. We include the N_i variable with the same meaning as in the CP model: the index of the period assigned to course i. We link N_i to x_{ij} as follows:

$$N_i = \sum_{j=1}^{n} x_{ij} j. \qquad i = 1, \ldots, m \tag{4.35}$$

The global constraints added to the MIQP model to form the CIP model are Constraints (4.21) and (4.23) from the CP model of the BACP (Fig. 4.3). The complete CIP model is the MIQP model in Fig. 4.4 plus Constraints (4.21), (4.23) and (4.35).

$$\min \quad \sigma \tag{4.27}$$

$$\text{s.t.} \quad \sigma \geq \sqrt{\frac{\sum_{j=1}^{n}(W_j - \mu)^2}{n}}, \tag{4.28}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, m \tag{4.29}$$

$$W_j = \sum_{i=1}^{m} x_{ij} L_i, \qquad j = 1, \ldots, n \tag{4.30}$$

$$MinCourses \leq \sum_{i=1}^{m} x_{ij} \leq MaxCourses, \qquad j = 1, \ldots, n \tag{4.31}$$

$$\sum_{j=1}^{n} x_{ij} j < \sum_{j=1}^{n} x_{\bar{i}j} j, \qquad \forall (i, \bar{i}) \in Prerequisites \tag{4.32}$$

$$x_{ij} \in \{0, 1\}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n \tag{4.33}$$

$$W_j \in [min\{L_i\}, \sum_{i=1}^{n} L_i\}]. \qquad j = 1, \ldots, n \tag{4.34}$$

Figure 4.4: The MIQP model for BACP.

4.4 Augmented Global Constraints

In Section 4.2.2 and Section 4.3.2 we proposed novel MIQP and CIP models for the NPAP and BACP. In this section, we focus on our more significant contribution: the augmented global constraints, which embed both inference techniques and constraint-specific linear relaxations and cutting planes. Specifically, we augment the MIQP models to include the QUADRATIC [23], GCC, and SPREAD global constraints within the CIP framework. For the first two constraints, the propagation algorithm, relaxation, and cutting planes are taken from the existing literature. For the SPREAD constraint, we implement the inference algorithm due to Schaus et al. [137] and develop novel relaxation and cutting plane techniques.

For each constraint, bounds propagation is used as the underlying inference algorithm in the solver used, SCIP, as it employs an interval representation of variable domains. As the functionality of global constraints in CIP is more general than in CP, we discuss each constraint in this section.

4.4.1 The Quadratic Constraint

The QUADRATIC constraint [23] is used to reason about constraints with quadratic terms and consists of a set of n variables $\{W_1, \ldots, W_n\}$, an $n \times n$ symmetric matrix $\boldsymbol{A}$, an n-dimensional vector $\boldsymbol{b}$, and scalars l and u. The representation is given as follows:

$$\texttt{quad}(\{W_1, \ldots, W_n\}, \boldsymbol{A}, \boldsymbol{b}, l, u),$$

where $\boldsymbol{A} \in \mathbb{Q}^{n \times n}$, $\boldsymbol{b} \in \mathbb{Q}^n$, $l \in \mathbb{Q}$ and $u \in \mathbb{Q}$. The constraint ensures the following condition:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} W_i W_j + \sum_{i=1}^{n} b_i x_i \in [l, u].$$

We use the existing QUADRATIC constraint in SCIP [23] with its default parameter values. It implements a number of problem solving techniques including bounds propagation, addition of linear relaxations, cutting plane generation, problem reformulation, and primal heuristics.

4.4.2 The Global Cardinality Constraint

The GCC constraint consists of a set of m variables $\{N_1, \ldots, N_m\}$, a set of n values $\{v_1, \ldots, v_n\}$, and a set of n pairs of values $[l_j, u_j]$, for each v_j. The constraint is satisfied if and only if each v_j is assigned at least l_j times and at most u_j times to the N_i variables. The representation is given as follows:

$$\texttt{cardinality}(\{N_1, ..., N_m\}, \{v_1, ..., v_n\}, \{[l_1, u_1], ..., [l_n, u_n]\}).$$

We use the bound consistency filtering algorithm due to Quimper et al. [128].

4.4.2.1 Relaxation

Linear relaxations have a central role in mixed integer programming. Given an MILP, which is, in general, NP-hard, the standard relaxation arises from ignoring the integrality constraints on the integer variables resulting in a polytime solvable linear program (LP). The LP plays numerous roles in search including providing a dual bound on a problem that is compared to the current best solution to prune search sub-trees in which no improving solution exists and in providing a basis for search heuristics [5, 125].

A substantial body of work has developed linear relaxations for global constraints (e.g., [79, 111, 129]). We implement an existing relaxation based on the MILP representation of GCC [79]. Using the notation for GCC introduced above, if we define an additional binary variable $x_{ij} = 1$ if $N_i = v_j$, an exact formulation of GCC is presented in Fig. 4.5. This formulation is used by the solver to form a linear relaxation of the GCC constraint by ignoring the integrality constraint on x_{ij} (i.e., Constraint (4.39)) and replacing it with $x_{ij} \in [0, 1]$.

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, \ldots, m \tag{4.36}$$

$$l_j \leq \sum_{i=1}^{m} x_{ij} \leq u_j, \qquad j = 1, \ldots, n \tag{4.37}$$

$$x_{ij} = 0, \qquad \forall i, j \notin D_{x_i} \tag{4.38}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i, j \tag{4.39}$$

$$N_i = \sum_{j=1}^{n} v_j x_{ij}. \qquad i = 1, \ldots, m \tag{4.40}$$

Figure 4.5: A MILP formulation of GCC [79].

Our CIP models already contain linear constraints that are identical to some of those in Fig. 4.5. Specifically, Constraint (4.11), (4.16), and (4.19) in the CIP model of the NPAP and Constraint (4.29), (4.33), and (4.35) in the CIP model of the BACP are identical Constraints (4.36), (4.39), and (4.40),

respectively. As Constraint (4.38) does not fix any x_{ij} variables in our problems, the only constraint we add to the CIP model is Constraint (4.37).

4.4.2.2 Cutting Planes

With the importance of the linear relaxation to MILP solving, techniques for improving it via the addition of cutting planes (i.e., implied linear constraints) have been developed [105]. We use the cutting planes for GCC due to Hooker [79].

For the CIP model we generate one inequality constraint using all x_i variables:

$$\sum_{i=1}^{n} p_i v_i \leq \sum_{j=1}^{m} N_j \leq \sum_{i=1}^{n} q_i v_i, \tag{4.41}$$

where

$$p_i = \min\Big\{u_i \; m - \sum_{j=1}^{i-1} p_j - \sum_{j=i+1}^{n} l_j\Big\}, \quad i = 1, \ldots, n \tag{4.42}$$

$$q_i = \min\Big\{u_i \; m - \sum_{j=i+1}^{n} q_j - \sum_{j=1}^{i-1} l_j\Big\}. \quad i = n, \ldots, 1. \tag{4.43}$$

Note that p_i and q_i are the maximum number of times that a value in N_j's domain, v_i, can be selected when, respectively, minimizing and maximizing the sum of the x_i variables, assuming, without loss of generality, that the v_is are ordered from smallest to largest. Similar inequalities including subsets of the x_i variables ranging in cardinality from 2 to $m-1$ are also valid and could result in a smaller search space. However, our preliminary results indicated that the large number of the constraints added to the problem results in a very large model, reducing search efficiency. Please see Hooker [79] for more details.

4.4.3 The Spread Constraint

The SPREAD constraint was first proposed by Pesant [124] to achieve the balancing of a set of values based on statistical concepts. A bound consistency algorithm was proposed in the same paper and a simplified and extended filtering algorithm was presented in Schaus et al. [137].

The SPREAD constraint enforces a given mean μ and maximum standard deviation σ among a set of n variables $\{W_1, \ldots, W_n\}$. It can be defined as follows:

$$\texttt{spread}(\{W_1, \ldots, W_n\}, \mu, \sigma).$$

The filtering algorithm reduces the domains of the W_j variables based on μ and σ in $O(n^2)$ time-complexity. Another algorithm filters values in the domain of σ based on domains of the variables W_j in quadratic time [137]. Both algorithms are implemented in the SPREAD constraint in this paper. The complete SPREAD constraint also propagates from W_j and σ to μ. However, since in our problem the total acuity load and number of nurses are fixed and known, the mean, μ, is always fixed to a single value. The filtering algorithm for the variable mean case is presented in our recent work [104].

4.4.3.1 Relaxation

A simple relaxation of the spread constraint is a single linear equality that enforces the mean among all the variables in the spread constraint. The constraint guarantees that the sum of all variables W_j is equal to $\mu \times n$, where n is the number of variables.

$$\sum_{j=1}^{n} W_j = \mu \times n. \tag{4.44}$$

4.4.3.2 Cutting Planes

When the objective is to minimize the standard deviation, the quadratic objective function can be formulated as follows:

$$\min \sigma = \sqrt{\frac{\sum_{j=1}^{m}(W_j - \mu)^2}{n}}, \tag{4.45}$$

which can be re-written in the following form:

$$\min \|\boldsymbol{\mu} - \boldsymbol{I}\boldsymbol{W}\|_2^2, \tag{4.46}$$

where $\boldsymbol{I} \in \mathbb{Z}^{n \times n}$ is the identity matrix, $\boldsymbol{\mu} = \mu \boldsymbol{e}$, $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{W} \in \mathbb{R}^n$ are n-dimensional vectors.

Suppose the optimal integer solution $\boldsymbol{W}^*$ to the above problem satisfies the following bound

$$\|\boldsymbol{\mu} - \boldsymbol{I}\boldsymbol{W}^*\|_2^2 < \beta, \tag{4.47}$$

where β is a constant. Geometrically, Equation (4.47) is a hyper-ellipsoid with center $\boldsymbol{\mu}$.

As seen in Section 3.2.3, we can apply the DEBS reasoning and derive the smallest hyper-rectangle whose edges are parallel to the axes of the coordinate system and that includes the hyper-ellipsoid. Recall that for the general form of the hyper-ellipsoid $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \beta$, the lower bound $\boldsymbol{l}$ and the upper bound $\boldsymbol{u}$ can be computed as follows:

$$u_k = \left\lfloor \sqrt{\beta} \left\| \boldsymbol{A}^{-\top} \boldsymbol{e}_k \right\|_2 + \boldsymbol{e}_k^\top \boldsymbol{A}^{-1} \boldsymbol{y} \right\rfloor,$$

$$l_k = \left\lceil -\sqrt{\beta} \left\| \boldsymbol{A}^{-\top} \boldsymbol{e}_k \right\|_2 + \boldsymbol{e}_k^\top \boldsymbol{A}^{-1} \boldsymbol{y} \right\rceil.$$

Since in our problem, $\boldsymbol{A} = \boldsymbol{I}$ and $\boldsymbol{y} = \boldsymbol{\mu}$, Equation (4.47) is actually a hyper-sphere with center $\boldsymbol{\mu}$ with the same lower bounds and upper bounds for all the variables. Thus, the computation of the lower bounds and the upper bounds can be simplified as follows:

$$u_k = \left\lfloor \sqrt{\beta} + \mu \right\rfloor, \tag{4.48}$$

$$l_k = \left\lceil -\sqrt{\beta} + \mu \right\rceil. \tag{4.49}$$

An upper bound of the standard deviation σ_{max} can be used to compute β through the following relationship, based on Equation (4.45) and (4.46):

$$\beta = \sqrt{n\sigma_{max}^2}. \tag{4.50}$$

Therefore, the cutting planes (4.48) and (4.49) can be computed and used to update the bounds on the W_j variables. In our implementation, these constraints are computed every time the upper bound of σ is improved. Since the constraints are globally feasible, the global bounds of the variables are updated if the new bounds are tighter than the current global bounds of the variables. In Section 3.2.3, we applied a similar approach to solving BQPs and ILS problems within CIP.

4.5 Branching Heuristics

It is well recognized in CP and MILP that the use of search heuristics can have substantial impact on problem solving performance [5, 18, 72, 125]. One simple way to influence search without implementing new heuristics is to modify the heuristic priority of variables. In SCIP, the branching priority of variables can be modified, allowing problem-specific knowledge to be incorporated into the default heuristics. A set of variables with higher branching priority will be branched on earlier in the search.

For the NPAP, we investigate two manipulations: 1) increasing the priority of the z_{jk} variables that assign nurses to zones and 2) increasing the branching priority of *both* the z_{jk} variables and the workload variables, W_j. For the BACP, we increase the branching priority of the workload variables. In all conditions, we assign maximum priority to all corresponding variables.

Increasing z_{jk} Priority. We choose to increase the branching priority of the z_{jk} variables based on the intuition that they have a significant effect on many other variables. Pryor & Chinneck [125] have shown that to quickly find a feasible solution in MILP, it is often desirable to branch on variables whose assignment results in substantial change to the linear relaxation. When a z_{jk} variable is set to 1, Constraint (4.12) immediately results in the $p-1$ other z_{ik} variables with $i = j$ being assigned to 0. Furthermore, through Constraint (4.14), fixing a z_{jk} variable to 0 leads to the assignment of m x_{ij} variables to 0. Therefore, whether branching up or down on the z_{jk} variables, we expect to see a substantial change in the linear relaxation.

Increasing W_j Priority. We choose to also increase the branching priority of the W_j variables due to their expected impact on the dual bound. A second intuition for heuristics in a MILP is to branch to quickly increase the dual bound (i.e., the lower bound in a minimization problem). It is often observed that a considerable amount of the effort in solving a problem is not in finding a solution but in proving its optimality [47]. As the primary method for such a proof is through pruning the sub-trees when the dual bound meets or exceeds the incumbent solution value, it is often useful to base branching heuristics on increasing the dual bound (e.g., [3]).

Given Constraint (4.10) of the NPAP and Constraint (4.28) of the BACP, branching on variables other than W_j will tend to lead to relaxed W_j values that are close μ, resulting in a dual bound that is close to 0. Branching on the W_j variables, in contrast, can more quickly increase the dual bound as the upper and lower bounds on other W_j variables must be changed due to Constraint (4.44).

For the NPAP, our experiments showed that only increasing the branching priority of the W_j variables without the z_{jk} led to poor performance due to difficulty in finding feasible solutions. We will return to this point in Section 4.7.2.

4.6 Experimental Results: NPAP

4.6.1 Setup

All experiments were performed on a Intel Core i7 3.40 GHz machine (in 64 bit mode) with 8GB memory running Red Hat Enterprise 6.2 with one thread. The software is CPLEX v12.6.3, SCIP v3.0.2, OSCAR v2.0.0 (using the bounds consistency SPREAD constraint), and ILOG CP v1.6 (using the domain consistency SPREAD constraint). The CPU time limit for each run on each problem instance is 7200 seconds.

For the CIP models, we compare the performance of the following nine model-heuristic combinations.

- The basic CIP is declaratively identical to the MIQP model (Fig. 4.2) but, when solved using default SCIP, incorporates the QUADRATIC constraint. We experiment with three models corresponding to the branching priorities: CIP (default), CIP_z (increased z_{jk} priority), and $CIP_{z,W}$ (increased priority for z_{jk} and W_j).
- The CIP model augmented to include only the constraint propagation of the GCC and SPREAD constraints is indicated by a superscript p (for propagation). There are, again, three models: CIP^p, CIP^p_z, and $CIP^p_{z,W}$ corresponding to the branching priorities.
- The full CIP model, indicated by superscript f includes constraint propagation, relaxation, and cutting planes of the QUADRATIC, GCC, and SPREAD constraints: CIP^f, CIP^f_z, and $CIP^f_{z,W}$.

4.6.2 Test sets

Following the methodology of Schaus et al. [139], we generated problem instances to closely resemble the original real world instances [116]. Specifically, the maximum acuity for a nurse is set to $MaxAcuity = 105$ and the maximum number of patients per nurse is $MaxPatients = 3$. We generate two zone ($p = 2$) instances with number of nurses, $n \in [9, 12]$, number of patients, $m \in [21, 30]$, and three zone ($p = 3$) instances with $n \in [13, 19]$, $m \in [36, 51]$. We generate 24 problem instances for both the two zone and three zone problems.

4.6.3 Results

Two Zone Problems. An overview of the results is given in Table 4.3. We report the arithmetic mean CPU time "arith", and the shifted geometric mean CPU time "geo" for finding and proving optimality on the 24 instances.[2] The arithmetic mean on the number nodes "Nodes" (number of backtracks "Bts" for CP) and the optimality gap "Opt gap" are also presented. We finally report the number of instances for which an optimal solution is found and proved "# opt" and the number of optimal solutions found "# opt found" without necessarily being proved.

The results of the new MIQP model demonstrate that state-of-the-art solver CPLEX is able to find optimal solutions to all problems,[3] and it is able to prove optimality in 22 of 24, however, with a substantially larger run-time as compared to the previously best known CP model in OSCAR. We note that the CP(DC) results are reported using a simple branching strategy that chooses a nurse

[2]The shifted geometric mean time is computed as follows: $\prod(t_i + s)^{1/n} - s$, where t_i is the actual CPU time, n is the number of instances, and s is chosen as 10. Using geometric mean can decrease the influence of outliers [3].

[3]In our previous publication [93] we used CPLEX v12.5, which is only able to prove optimality in 7 of 24.

Solver	Model	Time to opt (sec)		Nodes or Bts	Opt gap (%)	# opt	# opt found
		arith	geo				
OSCAR	CP(BC)	86.02	20.97	9972713	0	24	24
ILOG CP	CP(DC)	4121.75	2432.47	60283636	0	13	16
CPLEX	MIQP	1683.39	522.81	3133729	0	22	24
SCIP	CIP	1763.78	523.39	4041783	1	21	23
	CIP^p	2706.14	725.31	4825965	3	17	19
	CIP^f	2191.18	547.17	4028996	5	19	20
	CIP_z	1570.28	258.64	3790472	19	20	23
	CIP_z^p	1701.88	320.01	3324735	0	20	24
	CIP_z^f	1471.04	270.38	2508124	7	21	21
	$CIP_{z,W}$	119.40	37.97	210485	0	24	24
	$CIP_{z,W}^p$	99.75	34.20	132342	0	24	24
	$CIP_{z,W}^f$	75.70	28.46	130012	0	24	24

Table 4.1: Comparison of CP, MIQP, and 9 different CIP variations. For the CP solvers, "BC" denotes bounds consistency and "DC" denotes domain consistency. "Time to opt" is the time in seconds to find and prove optimality. We report both the arithmetic mean time "arith" and the shifted geometric mean time "geo". The optimality gap "Opt gap" is computed only for the problem instances where feasible solutions are found.

with the minimum domain size. We did not report the results using the branching rule that breaks symmetry dynamically because this strategy is not able to prove any of the instances to optimality when implemented in CP v1.6. Upon inspection of the backtracking patterns of CP v1.6, one possible explanation of this unexpectedly bad performance is that the backtracking decisions made by CP v1.6 might violate the dynamic symmetry breaking rule. In contrast, this situation does not happen in OSCAR. It is worth pointing out that comparing the performance of different CP solvers using different filtering algorithms is delicate, so we do not emphasize the relative performance of these two CP models. Nevertheless, it is clear that the branching rule has a significant impact on solving the NPAP.

Turning to the CIP results, the first set of our new CIP models (CIP, CIP^p, and CIP^f) show worse performance than MIQP: fewer problems solved to optimality with about 30% more nodes. The inclusion of constraint propagation in the GCC and SPREAD constraints *degrades* performance (CIP^p vs. CIP) both in terms of run-time and search tree size. The 20% larger search tree in particular is interesting and deserves more study. We speculate, given results below, that we may be observing a negative interaction between the constraint propagation and relaxation-based MILP-style search, indicating that we cannot necessarily expect improved performance from simply adding global constraint propagation to a MILP-style search. Comparing CIP^f to CIP shows similar run-times and tree sizes but worse solution quality for CIP^f. We believe that the gains from the global constraints are not large enough to out-weigh the increased computation they incur.

In the second set of CIP results, with increased branching priority for the z_{jk} variables, we see a substantial performance improvement in all models compared to the default heuristic. The inclusion of global constraints, whether just the propagation or the full model, results in smaller search trees by about 12% for CIP_z^p and 34% for CIP_z^f and better solution quality. However, the run-times are either about the same or are actually worse than the CIP_z model.

Finally, the most substantial gains arise from increasing the branching priority of both the z_{jk} variables and the W_j variables. About an order of magnitude improvement is seen compared to the

Solver	Model	Time to opt (sec)		Nodes or Bts	Opt gap (%)	# opt
		arith	geo			
OSCAR	CP(BC)	5384.11	2992.33	542901857	27	8
CPLEX	MIQP	6260.89	5345.15	11452100	12	5
SCIP	CIP^f_w	2433.56	1316.99	3031630	9	21

Table 4.2: Results of the three zone NPAP. All notations are the same as in Table 4.1. We do not report the number of optimal solutions found "# opt found" as we do not know the optimal solutions for all the instances. The optimality gap is computed as: opt gap $= (UB(\sigma) - LB(\sigma))/LB(\sigma) \times 100$, where $UB(\sigma)$ is the best upper bound and $LB(\sigma)$ represents the best lower bound found by our CIP approach at 7200 seconds, or the known optimal solution cost, if an instance is proved optimal.

previous CIP models. In addition, all the three CIP models outperform the MIQP model using CPLEX. The inclusion of global constraint propagation plus relaxation and cutting planes leads to clear gains with search tree sizes of almost 40% smaller than $CIP_{z,W}$. Furthermore, our best CIP model, $CIP^f_{z,W}$, solves all problems to optimality and outperforms the CP model in OSCAR in terms of arithmetic mean running time by 14%.[4]

Three Zone Problems. The results of the three zone problems are given in Table 4.2. Although both our best CIP model and the CP model in OSCAR solve the two zone problems very quickly, they cannot solve all of the three zone problems to optimality within the time limit. However, our best CIP model greatly outperforms the CP model, proving 2.6 times more instances to optimality with a much smaller optimality gap. We believe this is the first model of any type that has been able to improve on the performance of the CP state-of-the-art. Recall that the three zone problems not only has one more zone but the number of patients and nurses are also substantially increased. Our results suggest that the CP approach is not able to scale as well as the CIP approach.

4.6.4 Discussion

Our experimental results have demonstrated that the hybridization of CP and MIQP techniques within the framework of CIP results in a new state-of-the-art for the load balancing NPAP.

Primal Bounds and Dual Bounds. Fig. 4.6 plots the evolution of the mean primal and dual bounds over time for the three CIP models of the two zone problems. For each instance, the primal, p, and dual, d, gaps are computed as follows: $p = (UB(\sigma) - \sigma^*)/\sigma^* \times 100$ and $d = (LB(\sigma) - \sigma^*)/\sigma^* \times 100$, where σ^* is the known optimal solution cost and $UB(\sigma)$ and $LB(\sigma)$ respectively represent the best upper and lower bounds at a given point in the search.

Consistent with our intuitions in Section 4.5, $CIP^f_{z,w}$ delivers tight primal and dual bounds, though it is more clearly dominant in the latter. However, the results of CIP^f_z contradict our expectations that increasing the priority of the z_{jk} variables alone would lead to strong primal bounds. We see the opposite, as CIP^f_z dominates CIP^f in terms of the dual bound while performing much worse on the primal bound. Interestingly, experiments that only increasing the priority of the W_j variables (not included here) do match our intuitions: the model performs poorly, timing out without finding *feasible*

[4]In our previous publication [93], all the three CIP models solve all problems to optimality from 7 to 10 times faster than the CP model implemented in COMET v2.1.1.

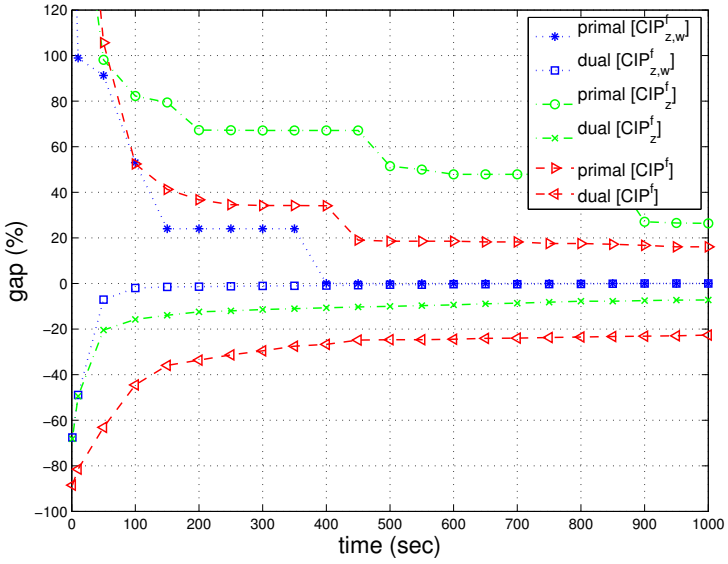


Figure 4.6: Comparison of the primal and dual gaps of CIP^f, CIP^f_z and $CIP^f_{z,w}$.

solutions to 5 problem instances. More detailed experimentation is needed to understand the reasons behind the impact of the search heuristics and, in particular, why $CIP^f_{z,w}$ performs so well.

The Impact of Global Constraints. Global constraints play a primary role in CP, forming the central object in both modeling and solving. Building on this role and the substantial work over the past 15 years on the hybridization of CP and MIP solving techniques [151], we are interested in exploring the concept of a global constraint as a richer object in the search process, exploiting its structure to do more than domain pruning [17].

Several works have shown the success on the pursuing of this direction. For example, global constraints can provide heuristic information [123], generate SAT clauses [141] and cutting planes [20], detect independent sub-problems [74], and decompose constraints and fix or remove variables [76].

Though we have developed a new state-of-the-art hybrid model for the NPAP and demonstrated the reduction in search effort (in both time and nodes) from the integration of constraint propagation, linear relaxation, and cutting planes within global constraints (i.e., compare CIP_z with CIP^f_z and $CIP_{z,w}$ with $CIP^f_{z,w}$), the importance of augmented global constraints to our results should not be overstated. Without the use of the branching priorities on z_{jk} and W_j variables, none of the CIP models are able to match the performance of the CP model.[5] Therefore, it appears that the primary reason for the strong performance of the new state-of-the-art is the branching priorities and, at best, the *interaction* of the branching priorities with the augmented global constraints. Furthermore, as the comparison of the number of nodes of CIP and CIP^p indicate, the simple addition of global constraint propagation to a MILP-style search does not necessarily result in improved performance: a more nuanced understanding of the interactions is needed.

[5]The CP model also uses a problem-specific heuristic and so we believe the direct comparison of it with the $CIP^f_{z,w}$ is justified.

Solver	Model	Time to opt (sec)		Nodes or Bts	Opt gap (%)	# opt	# opt found
		arith	geo				
OSCAR	CP(BC)	0.04	0.04	1261	0	48	48
ILOG CP	CP(DC)	0.03	0.03	145	0	48	48
CPLEX	MIQP	2.49	1.80	6673	0	48	48
SCIP	CIP	2.43	1.92	7935	0	48	48
	CIP^p	1.95	1.78	4800	0	48	48
	CIP^f	1.51	1.34	4793	0	48	48
	CIP_w	0.84	0.82	2812	0	48	48
	CIP_w^p	1.12	1.06	2899	0	48	48
	CIP_w^f	0.87	0.81	3159	0	48	48

Table 4.3: Results of the single zone NPAP. All notations are the same as in Table 4.1.

Comparison of CP(BC) and CP(DC). It is surprising that the CP model using the domain consistency filtering algorithm performs significantly worse than that using the bounds consistency algorithm. Recall that dynamic symmetry breaking rule exploits the equivalence among all nurses who have not yet been assigned a patient. The search dynamically breaks the value symmetries originating from the nurse interchangeability by considering the already assigned nurses and at most one additional nurse without any assigned patient. Specifically, let the value $maxUsed$ be the maximal index of a nurse already assigned to a patient. We only consider the nurses with indices less than or equal to $maxUsed+1$, i.e., nurse $maxUsed+1$ currently has no patient. In CP v1.6, a patient might be assigned to a nurse that is greater than $maxUsed+1$ during backtracking, thus violating the dynamic symmetry breaking branching rule. We speculate that this weakens the strength of symmetry breaking and leads to inferior overall performance.

Single Zone Problems. For the sake of completeness, we also experiment with the single zone problems, which are only concerned with assigning patients to nurses. We use the same 24 instances as those in Section 4.6.2, but with the number of nurses per zone precomputed using the heuristic in Schaus et al. [139], resulting in 48 single zone problems. Table 4.3 shows that all approaches are able to solve the single zone NPAP very quickly. The CP approach using the domain consistency filtering algorithm performs the best, which is consistent with the results reported in [121, 122]. All approaches are able to prove optimality in less than 3 seconds. In a real world setting, the performance difference might be insignificant.

Comparison of CIP and Decomposition Method. We end this section with a note on the relative performance between our best CIP model and the exact CP-based decomposition method proposed by Pesant [122] recently. Pesant's method decomposes the multi-zone problems into independent single zone problems that can be solved very efficiently with CP, as shown in Table 4.3. The average running time for solving the three zone problems with the decomposition method is about three order of magnitudes faster than our CIP approach, as expected. However, for problems that are naturally non-decomposable, e.g., problems with inter-zone constraints, our CIP approach remains the state of the art.

4.7 Experimental Results: BACP

The experimental setup is the same as the one used for the NPAP in Section 4.6.1.

4.7.1 Test sets

We use the benchmark instances from Schaus [136], which consist of 100 instances. These instances were generated randomly with the academic loads of each course between 1 and 5. There are 66 courses, 12 semesters, and 50 prerequisite constraints that define the pairwise precedence relationships between two courses. The number of courses per semester is constrained between 5 and 7.

4.7.2 Results and Discussion

An overview of the results is given in Table 4.4. Our new MIQP model and augmented CIP model with branching heuristics both outperform the best known CP models by at least two orders of magnitude. In addition, our results show that the latest version of CPLEX performs the best, proving all instances to optimality in about 2 seconds. We attribute this strong performance to the recent improvement on the MIQP techniques in CPLEX [31].

The results of the MIQP model and the CP model show opposite results to those of the NPAP, i.e., the MIQP model performs significantly better than the CP model for the BACP. Recall that in the (full) NPAP, both the nurse-to-zone and the nurse-to-patient decisions have to be made. In the BACP, similar to the one zone NPAP, only one decision is required, which is the course-to-period assignment. We observe a very rapid dual bound improvement for the BACP and we believe that this accounts for the superior performance of CPLEX.

The first set of CIP models (CIP, CIP^p, and CIP^f) show that the inclusion of constraint propagation in the GCC and SPREAD constraints does not affect the performance (CIP^p vs. CIP) in terms of run-time in a substantial way. Comparing CIP^f to CIP shows only a slight improvement in terms of number of instances that are proved optimal.

With increased branching priority for the W_j variables, we see a significant performance improvement in the second set of CIP results. As observed in the NPAP, the inclusion of global constraint propagation plus relaxation and cutting planes leads to clear gains in terms of search tree sizes and running time. It is interesting to observe that while all the CIP models are able to find the optimal solutions, only the models with the branching heuristic are able to prove optimality efficiently.

BACP Without the Cardinality Constraint. We also experiment with a variation of the BACP in the literature [114, 136], which ignores the cardinality constraint that enforces the number of courses per period. Consistent with previous results [121], the CP approaches perform substantially better on this problem variation than the original problem. While the same conclusion can be drawn for the MIQP and CIP approaches, Table 4.5 shows that MIQP and CIP are much less sensitive to the existence of the cardinality constraints: It is unlikely to make a significant portion of the non-solvable problems solvable by removing the cardinality constraint from the problem formulation.

While our best CIP model performs worse (within one order of magnitude) than the MIQP model in CPLEX for the BACP, it is worth pointing out that CPLEX is usually considered significantly faster than SCIP [113], the underlying solver of our CIP models. In our case, CPLEX is at least 3 orders of

Solver	Model	Time to opt (sec)		Nodes or Bts	Opt gap (%)	# opt	# opt found
		arith	geo				
OSCAR	CP(BC)	2668.14	264.56	10015243	13	69	69
ILOG CP	CP(DC)	2246.35	173.43	6824036	18	72	72
CPLEX	MIQP	2.15	1.92	674	0	100	100
SCIP	CIP	6336.59	3422.98	9504397	0	12	100
	CIP^p	6336.76	3456.30	4967846	0	12	100
	CIP^f	6272.14	3340.73	5999422	0	13	100
	CIP_w	868.82	21.85	2100346	0	88	100
	CIP_w^p	34.98	24.57	6712	0	100	100
	CIP_w^f	12.72	11.61	1518	0	100	100

Table 4.4: Results of the BACP. All notations are the same as in Table 4.1.

Solver	Model	Time to opt (sec)		Nodes or Bts	Opt gap (%)	# opt	# opt found
		arith	geo				
OSCAR	CP(BC)	217.69	3.14	139731	1	97	97
ILOG CP	CP(DC)	147.31	3.10	47542	1	98	98
CPLEX	MIQP	1.25	1.18	621	0	100	100
SCIP	CIP	6172.13	2968.95	6408372	0	12	100
	CIP^p	6171.98	2939.95	8292744	0	12	100
	CIP^f	6172.05	2949.78	7620029	0	13	100
	CIP_w	869.87	23.85	1572461	0	88	100
	CIP_w^p	5.57	5.35	1098	0	100	100
	CIP_w^f	5.51	5.29	1089	0	100	100

Table 4.5: Results of the BACP without the cardinality constraint. All notations are the same as in Table 4.1.

magnitude faster than the default CIP model. Our branching heuristics and global constraints therefore have substantially improved the default CIP performance.

4.8 Conclusion

In this chapter, we developed a series of novel MIQP and CIP models for the load balancing NPAP and the BACP. These problems are challenging and have been addressed with mixed integer linear programming and constraint programming, with the latter representing the state-of-the-art.

Our approach focused on the integration of augmented global constraints into the CIP model. In addition to constraint propagation, the global constraints implemented constraint-specific linear relaxations and cutting plane generation. Building on the existing work on the QUADRATIC [23], GCC [79, 128], and SPREAD [137] constraints, we introduced a linear relaxation and cutting planes for the latter.

For the NPAP, our empirical results demonstrate that the default CIP model performs slightly worse than the MIQP model implemented in CPLEX and does not compete with the non-decomposition based CP model. To facilitate the search process, we propose problem-specific branching priorities, which greatly improve the CIP models. Results show that our best CIP model significantly outperforms the previous best known CP model, especially when the scale of the problem is increased. For the BACP, both our new MIQP model and our augmented CIP model with branching heuristics outperform

the best known CP models, with the MIQP model using CPLEX performing the best. As for the relative performance among the CIP models, our results show the same trend as those observed in the NPAP, supporting the effectiveness of the augmented global constraints. In summary, our CIP approach demonstrates how techniques from CP and MIP can benefit from each other and achieve the state-of-the-art or near state-of-the-art for both the NPAP and BACP.

For future work, we would like to investigate other possibilities of using the information from global constraints to enhance the search. We propose two potential directions for further exploration: constraint-specific lazy constraint generation [120] and constraint-specific based branching.

Lazy constraint (clause) generation [120] is a powerful technique to reduce the search effort in CP by recording the reasoning that leads to a failure and creating an implication graph that can be used to infer no-goods. To generate constraint-specific no-goods, a global constraint must be extended to explain itself. For example, Downing [53] has shown how various propagation algorithms of the ALLDIFFERENT constraint can be extended to explain themselves and generate no-goods in a CP solver. It has been shown that learning the no-goods for problems involving the ALLDIFFERENT constraint often lead to tremendous reduction in running time. Results show that the combination of the ALLDIFFERENT propagators with their explanation algorithms result in a state-of-the-art CP approach for several problems involving the all-different structure. While lazy constraint generation has been proved to be a useful technique for the ALLDIFFERENT and CUMULATIVE constraints [75, 141], the explanation algorithms have not been developed for many other well-known global constraints, including the GCC and the SPREAD constraint. As a starting point, we would like to generate explanations for the infeasibility that result from the propagation of these two constraints to reduce search effort.

A global constraint might also provide useful information for branching decisions. For example, the *counting-based search* [123] examines each constraint and analyzes the frequency of a given variable-value assignment in the solutions of the constraint. Such information can be used to design branching heuristics that guide the search toward the regions that are more likely to contain solutions. In addition to solution density, a global constraint may provide other structure information to enhance the search. For example, the SPREAD constraint achieves its minimum standard deviation when all of its variables take values of its mean. Intuitively, the standard deviation increases when the variables take values farther from the mean value. During our preliminary investigation, we observed that a value branching rule developed w.r.t. the mean value can reduce the number of branches by about 20% compared to our best CIP approach for both the NPAP and the BACP. We believe that such constraint-specific branching strategies can be applied successfully to problems where global constraints play a major role in solving the problem.

As a final note, although we investigated the idea of the augmented global constraint in the CIP paradigm, applicable techniques can be chosen selectively for CP and MIP alone, e.g., a MIP solver can easily adapt all the techniques from a augmented global constraint except the filtering algorithm.

In the next chapter, we build on the work in Chapter 3 and Chapter 4 and develop a CP approach to solve the strictly convex integer quadratically-constrained problems, which expands upon the type of problems that were previously considered to the much more general setting.

Chapter 5

Constraint Programming for Strictly Convex Integer Quadratically Constrained Problems

In Chapter 3 and Chapter 4 we proposed a number of hybrid approaches that combine mixed integer programming (MIP), constraint programming (CP) and discrete ellipsoid-based search (DEBS) to solve special cases of the strictly convex integer quadratically constrained problem. In this chapter, we develop a CP approach that builds upon the previous chapters to solve a much broader class of problems with general strictly convex quadratic constraints, allowing the incorporation of any side constraints that fit in the CP framework. Inspired by the geometric reasoning exploited in DEBS, we strengthen the key aspects of the DEBS approach and implement them as combination of a global constraint and variable/value ordering heuristics in IBM ILOG CP Optimizer.

5.1 Introduction

The strictly convex integer quadratically constrained problem (IQCP) is an important sub-class of mixed integer nonlinear programming (MINLP) problems where the objective and/or some constraints are strictly convex quadratic functions. As mentioned in the previous chapters, the IQCP is NP-hard [148] and arises in a number of applications including algorithmic number theory [13], cryptography [119], global positioning systems [146], wireless communications [6], and scheduling [94]. Despite the long history of CP, the techniques to solve quadratically constrained problems have not receive much attention. There are only a few dedicated global constraints that reason about quadratic terms. For example, the Spread constraint [124] enforces the standard quadratic relationship amongst a set of variables, their mean, and their standard deviation. General quadratic constraints [52, 96] can be applied to both convex and nonconvex quadratic functions. However, they do not exploit the strictly convex nature of the IQCP.

We showed in Section 2.2.3 that strictly convex IQCPs can be formulated as integer least squares (ILS) problems and solved with DEBS. From a CP perspective, DEBS can be understood as a form of CP search. First, the search strategy uses a static variable ordering heuristic and a dynamic value ordering

heuristic based on the structure of the ellipsoid. Second, the geometry of the ellipsoid induces an interval domain for each variable. As a result, DEBS is essentially the enumeration of these domains under the prescribed variable and value orderings with some bounds pruning based on the radius of the hyper-ellipsoid. DEBS was originally formulated to solve only three types of ILS problems: unconstrained, box-constrained, and ellipsoid-constrained [42, 43, 89, 92]. In Section 3.3, we extended DEBS for ILS problems with general linear constraints.

In this chapter, we develop techniques that are both powerful and cover a broad range of problems. We introduce two novel techniques, inspired by DEBS, for solving strictly convex IQCPs with constraint programming. First, we propose the ELLIPSOID constraint, a global constraint that filters variable domains with respect to strictly convex quadratic functions. We derive a direct quadratically constrained programming (QCP) formulation that achieves bounds consistency (BC), and two light-weight filtering algorithms that do not guarantee BC. Though it is natural to consider integer domains in CP, our filtering algorithm can be applied to variables with real domains, broadening its application to, for example, mixed integer programming solvers. Second, we propose a pair of variable/value selection rules. We implement the filtering algorithms and the branching heuristics in IBM ILOG CP Optimizer and experiment with five problem classes, showing orders of magnitude improvement compared to the default CP Optimizer. We then compare our new CP approach with the best known algorithms on the same problem sets. Our results demonstrate that the new CP approach is competitive to the best known approaches, and, for some problem classes, establishes a new state of the art.

5.1.1 Contributions

The contributions of this chapter are as follows:

- To the best of our knowledge, this is the first work that solves the strictly convex IQCPs with CP. We make a strong connection between CP and MINLP and bring a problem often seen as intractable to the CP community, incorporating it effectively in the framework of global constraints and branching strategies. We believe that this work can attract the OR community to the CP way of thinking and encourage further exploration of using CP as a basis for solving MINLPs.

- Our novel approach brings CP within an order of magnitude of the state-of-the-art techniques for the IQCPs tested. When compared to non-problem specific solvers, our CP approach even achieves the state of the art on some problem types. The use of our new global constraint and branching rules, as opposed to hand-crafted, specialized algorithms, provides more flexibility and fits well in the "model and solve" framework, allowing practitioners to utilize our techniques effortlessly.

The work in this chapter is based on the publication [91].

5.1.2 Organization

The rest of the chapter is organized as follows. We give the necessary background in Section 5.2. In Section 5.3 we define of our new global constraint. Section 5.4 and 5.5 present the filtering algorithms and the branching heuristics. Section 5.6 provides computational results and discussions. We conclude in Section 5.7.

5.2 Background

5.2.1 The Strictly Convex Integer Quadratically Constrained Problem (IQCP)

The general IQCP problem has the following form:

$$\min_{\boldsymbol{x}\in\mathcal{C}} \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x},$$

$$\mathcal{C} = \left\{ \boldsymbol{x} \in \mathbb{Z}^n : \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{M}_k \boldsymbol{x} + \boldsymbol{c}_k^\top \boldsymbol{x} \leq b_k, \forall k = 1,\ldots,m,\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n \right\}.$$

The IQCP is strictly convex if the quadratic matrices $\boldsymbol{H}$, $\boldsymbol{M}_k, \forall k$ are symmetric positive definite [66]. As the above form suggests, quadratic formulations can exist in the form of an objective function and/or as constraints.

In operations research, the common generic approaches to solving IQCPs exactly are the use of MINLP solvers such as BARON [134, 145], BONMIN [29] and ANTIGONE [112], and the application of MIP solvers such as CPLEX and Gurobi which have been extended to reason about quadratic constraints [37] (See Section 2.3.1.2 for details on MIP solving techniques for IQCPs). Another generic approach is semi-definite programming (SDP) based branch-and-bound [87]. The available SDP solvers, e.g., BiqCrunch [88], only solve problems with binary variables as opposed to general integer variables.

5.2.2 Discrete Ellipsoid-based Search (DEBS)

We briefly review the discrete ellipsoid-based search (DEBS) method for solving ILS problems. The DEBS method consists of two phases: reduction and search. The reduction is a preprocessing step that transforms $\boldsymbol{A}$ to an upper triangular matrix $\boldsymbol{R}$ using the QRZ factorization [42]:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \rightarrow \min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \tag{5.1}$$

where $\bar{\boldsymbol{y}} = \boldsymbol{Q}^\top \boldsymbol{y}$, $\boldsymbol{z} = \boldsymbol{Z}^{-1}\boldsymbol{x}$, $\boldsymbol{Q}$ is orthogonal, and $\boldsymbol{Z}$ is unimodular. The diagonal entries of $\boldsymbol{R}$ are approximately[1] ordered in non-decreasing order: $|r_{ii}| \leq |r_{i+1,i+1}|$. This ordering has been shown to increase the efficiency of the DEBS search by reducing the branching factor at the top of the search tree [43]. As we argue in Section 5.5 below, this ordering is approximately equivalent to a static smallest domain first variable ordering.

Suppose the optimal solution $\boldsymbol{z}^*$ satisfies $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}^*\|_2^2 < \beta$, or equivalently $\sum_{k=1}^{n}(\bar{y_k} - \sum_{j=k}^{n} r_{kj} z_j)^2 < \beta$, where β is a constant that can be obtained by substituting any feasible integer solution to equation (5.1). This expression defines a hyper-ellipsoid with center $\boldsymbol{R}^{-1}\bar{\boldsymbol{y}}$. The search, then, systematically enumerates all the integer points in the bounded hyper-ellipsoid [140]. When an incumbent, i.e., new upper bound on β, is found, the hyper-ellipsoid is contracted resulting in reduction of the bounds of the decision variables. After the optimal solution $\boldsymbol{z}^*$ to the reduced problem (right hand side of (5.1)) is found, the optimal solution, $\boldsymbol{x}^*$, to the original problem (left hand side of (5.1)) can be recovered with the relationship $\boldsymbol{x}^* = \boldsymbol{Z}\boldsymbol{z}^*$.

See Section 2.3.4 for more details on DEBS.

[1] Depending on the data, it is sometimes not possible to transform a matrix to exactly achieve this ordering [33].

5.3 The Ellipsoid Constraint

We propose the ellipsoid constraint to reason about convex quadratic functions. It consists of a set of n variables $\{x_1, \ldots, x_n\}$, an $n \times n$ matrix $\boldsymbol{A}$ with full column rank, an n-dimensional vector $\boldsymbol{y}$, and a constant β. The definition is given as follows:

$$\texttt{ellipsoid}(\{x_1, \ldots, x_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta),$$

where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\boldsymbol{y} \in \mathbb{R}^n$, $\beta \in \mathbb{R}$. The constraint ensures the following condition:

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \beta. \tag{5.2}$$

Geometrically, the above expression defines a hyper-ellipsoid with center $\boldsymbol{A}^{-1}\boldsymbol{y}$. Equivalently, (5.2) can be written in its standard convex quadratic constraint form as

$$\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x} \leq \bar{\beta}, \tag{5.3}$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, $\boldsymbol{f} \in \mathbb{R}^n$ is a vector, and $\bar{\beta} = (\beta - \boldsymbol{y}^\top \boldsymbol{y})/2$. The transformation is obtained with the relationships $\boldsymbol{H} = \boldsymbol{A}^\top \boldsymbol{A}$ and $\boldsymbol{f} = -\boldsymbol{y}^\top \boldsymbol{A}$.

The ellipsoid constraint can be applied to any formulation with a strictly convex quadratic function. For example, consider the following objective function: $\min \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x} + \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{H}_1 \boldsymbol{y} + \boldsymbol{f}_1^\top \boldsymbol{y}$, where only $\boldsymbol{H}$ is symmetric positive definite. We can still apply the ellipsoid constraint to the first half of the objective function: $\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x}$, even if the second part is not strictly convex.

5.4 Filtering Algorithms for the Ellipsoid Constraint

In this section, we present a number of filtering algorithms that achieve or approximate bounds consistency of the ellipsoid constraint.

Let x_j be a finite-domain variable, $D(x_j)$ be the domain of x_j, which is a set of ordered values that can be assigned to x_j, and $I_D(x_j) = [l_j, u_j]$ be the interval domain of x_j.

Definition 5.4.1. An ellipsoid constraint is bounds consistent [50] with respect to domains $D(x_j)$ if for all $j \in 1, \ldots, n$ and each value $v_j \in \{l_j, u_j\}$, there exists values $v_i \in I_D(x_i)$ for all $i \in \{1, \ldots, n\} \setminus \{j\}$ such that $\texttt{ellipsoid}(\{x_1 = v_1, \ldots, x_n = v_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta)$ holds.

In the 2D example shown in Fig. 5.1, the rectangle (in the shaded area) contains all the integer values that are bounds consistent. For all these integer value in each dimension, we can find a value in the other dimension such that the ellipsoid constraint holds. For each integer value no longer in a variable domain, there does not exist a value in the other variable domain that forms a point inside the ellipse. Therefore, the ellipsoid constraint (on the two variables) is bounds consistent.

5.4.1 A Direct Quadratically Constrained Programming (QCP) Formulation

Achieving bounds consistency for a variable x_j is equivalent to finding the lower bound l_j^{BC} and upper bound u_j^{BC} for x_j, given the ellipsoid constraint and the current bounds of the variables. Assume that

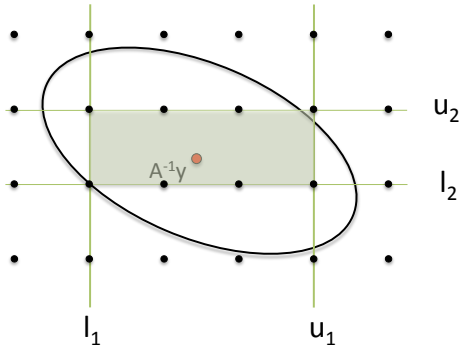


Figure 5.1: A 2D example that shows the tangent box of the ellipsoid and bounds consistency of the ELLIPSOID constraint.

no further reduction can be inferred on the domains of $x_i, \forall i \neq j$, the mathematical model for achieving bounds consistency for x_j can be defined as follows:

$$l_j^{BC} = \min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{e}_j^\top \boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \sqrt{\beta},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \tag{5.4}$$

$$u_j^{BC} = \max_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{e}_j^\top \boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \sqrt{\beta},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \tag{5.5}$$

Note that $\boldsymbol{e}_j \in \mathbb{R}^n$ is the unit vector in the j-th direction, i.e., the j-th column of an identity matrix with size n. The problems (5.4) and (5.5) are quadratically constrained programming (QCP) optimization problems, which can be solved with QCP solvers such as CPLEX. However, it is computationally expensive, since at least $2n$ QCPs have to be solved in each iteration.

This approach is essentially a QCP version of optimization-based bound tightening (OBBT) as used in the MINLP literature [64]. While typically performed with linear constraints, OBBT is often only done at the root node of the search tree as it is too expensive to perform at every node. Convex QCPs can be solved in polynomial time using iterative approaches such as the interior point method [118].

5.4.2 Axis-Aligned Tangent Box Filtering (BOX)

The simplest way to tighten the domains of the variables is to compute the tangent box of the hyper-ellipsoid defined in Equation (5.2), where the edges of the box are parallel to the axes of the coordinate system. As a 2D example, the dotted box in Fig. 5.1 shows the tangent box. The lower bound $\boldsymbol{l}^b$ and the upper bound $\boldsymbol{u}^b$ that define the tangent box can be computed by solving the following problems:

$$l_j^b = \min_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{e}_j^\top \boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \sqrt{\beta}, \tag{5.6}$$

$$u_j^b = \max_{\boldsymbol{x} \in \mathbb{R}^n} \boldsymbol{e}_j^\top \boldsymbol{x} \quad \text{subject to } \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \sqrt{\beta}. \tag{5.7}$$

Chang & Golub [42] proposed an efficient way to solve the above problems. We first solve the problem in (5.7) for u_j^b, and the lower bound l_j^b can be obtained by using the symmetric property of an ellipsoid.

Let $\boldsymbol{p} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}$, the problem (5.7) becomes

$$\begin{aligned} u_j^b &= \max_{\boldsymbol{p}} \boldsymbol{e}_j^\top \boldsymbol{A}^{-1}(\boldsymbol{p} + \boldsymbol{y}) \\ &= \max_{\boldsymbol{p}} \boldsymbol{e}_j^\top \boldsymbol{A}^{-1}\boldsymbol{p} + \boldsymbol{e}_j^\top \boldsymbol{A}^{-1}\boldsymbol{y} \quad \text{subject to } \|\boldsymbol{p}\|_2 \leq \sqrt{\beta}. \end{aligned} \tag{5.8}$$

By the Cauchy-Schwarz inequality, we have

$$\boldsymbol{e}_j^\top \boldsymbol{A}^{-1}\boldsymbol{p} \leq \left\|\boldsymbol{A}^{-\top}\boldsymbol{e}_j\right\|_2 \|\boldsymbol{p}\|_2 \leq \left\|\boldsymbol{A}^{-\top}\boldsymbol{e}_j\right\| \sqrt{\beta}.$$

The first inequality becomes an equality if and only if $\boldsymbol{p} = c\boldsymbol{A}^{-\top}\boldsymbol{e}_j$ for some non-negative scalar c. The second inequality becomes equality if and only if $\|\boldsymbol{p}\|_2 = \sqrt{\beta}$. Therefore, $\boldsymbol{p}$ is the minimizer for (5.8) when $\boldsymbol{p} = \sqrt{\beta}\boldsymbol{A}^{-\top}\boldsymbol{e}_j / \left\|\boldsymbol{A}^{-\top}\boldsymbol{e}_j\right\|_2$. Substituting $\boldsymbol{p}$ into (5.8), we have

$$u_j^b = \sqrt{\beta}\left\|\boldsymbol{A}^{-\top}\boldsymbol{e}_j\right\|_2 + \boldsymbol{e}_j^\top \boldsymbol{A}^{-1}\boldsymbol{y}. \tag{5.9}$$

From the symmetry property of an ellipsoid, we have

$$l_j^b = -\sqrt{\beta}\left\|\boldsymbol{A}^{-\top}\boldsymbol{e}_j\right\|_2 + \boldsymbol{e}_j^\top \boldsymbol{A}^{-1}\boldsymbol{y}. \tag{5.10}$$

5.4.2.1 Computing the Reduced Intersecting Ellipsoid $\mathcal{E}_\mathcal{F}$

When a variable is fixed during the search, e.g., $x_i = v_i$, the dimension of the ellipsoid is reduced by one. Geometrically, we need to find the one-dimension-smaller ellipsoid that intersects at $x_i = v_i$ and $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \beta$. We propose a general way to compute the reduced intersecting ellipsoid with any number of fixed variables.

Let $\mathcal{F}$ be the set of the variables that are fixed and let $\tilde{\boldsymbol{A}} = [\boldsymbol{A}_j], \forall j \neq \mathcal{F}$, $\bar{\boldsymbol{y}} = \boldsymbol{y} - \sum_{i\in\mathcal{F}} \boldsymbol{A}_i v_i$ and the QR factorization of $\tilde{\boldsymbol{A}}$ be $\tilde{\boldsymbol{A}} = \begin{bmatrix}\tilde{\boldsymbol{Q}}_1 & \tilde{\boldsymbol{Q}}_2\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{R}} \\ \boldsymbol{0}\end{bmatrix}$, the reduced intersecting ellipsoid $\mathcal{E}_\mathcal{F}$ can be computed as follows:

$$\mathcal{E}_\mathcal{F} = \left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}\right\|_2^2 \leq \tilde{\beta}, \tag{5.11}$$

where $\tilde{x}$ is the vector of the unknown variables of the reduced ellipsoid, $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{Q}}_1^\top \bar{\boldsymbol{y}}$ and $\tilde{\beta} = \beta - \|\bar{\boldsymbol{y}}\|_2^2 + \|\tilde{\boldsymbol{y}}\|_2^2$. The derivations on $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{y}}$ are straightforward so we only explain $\tilde{\beta}$ as follows. We know that

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \left\|\bar{\boldsymbol{y}} - \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}\right\|_2^2 = \left\|\begin{bmatrix}\tilde{\boldsymbol{Q}}_1^\top \\ \tilde{\boldsymbol{Q}}_2^\top\end{bmatrix}\bar{y} - \begin{bmatrix}\tilde{\boldsymbol{R}} \\ \boldsymbol{0}\end{bmatrix}\tilde{\boldsymbol{x}}\right\|_2^2.$$

It follows that

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 - \left\|\tilde{\boldsymbol{Q}}_1^\top \bar{\boldsymbol{y}} - \tilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}\right\|_2^2 = \left\|\tilde{\boldsymbol{Q}}_2^\top \bar{\boldsymbol{y}}\right\|_2^2 = \|\bar{\boldsymbol{y}}\|_2^2 - \left\|\tilde{\boldsymbol{Q}}_1^\top \bar{\boldsymbol{y}}\right\|_2^2.$$

We assume that β and $\tilde{\beta}$ constraints are satisfied at equality as this corresponds to the largest ellipsoids defined by our inequalities and therefore ensures that no valid values are pruned. Since $\beta = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$ and $\tilde{\beta} = \left\|\tilde{\boldsymbol{Q}}_1^\top \bar{\boldsymbol{y}} - \tilde{\boldsymbol{R}}\tilde{\boldsymbol{x}}\right\|_2^2$, we have $\tilde{\beta} = \beta - \|\bar{\boldsymbol{y}}\|_2^2 + \|\tilde{\boldsymbol{y}}\|_2^2$.

At each node of the tree, we can first compute the reduced ellipsoid $\mathcal{E}_F$ w.r.t. the variables that are already fixed with Equation (5.11), then apply Equations (5.9) and (5.10) to compute the axis-aligned

tangent box. The algorithm propagates when variables are instantiated or β is reduced. Because our CP search instantiates a variable at each node, the propagation is active at each node.

5.4.2.2 Complexity of the BOX Filtering Algorithm

The filtering algorithm reduces the domains of all the variables based on β with $O(n^3)$ time-complexity. First, computing the reduced ellipsoid takes $O(n^3)$, as the QR factorization is required. Second, computing the tangent box takes $O(n^3)$, as the complexity is dominated by computing the matrix inverse $\boldsymbol{A}^{-1}$.[2] Therefore the total time complexity for filtering the domain for all the variable is $O(n^3)$.

5.4.2.3 Improving the Complexity

We end this section with a note on improving the complexity of the filtering algorithm. Let $(l_j^F)^b$ be the lower bound of the reduced ellipsoid, we have $(l_j^F)^b = -\sqrt{\tilde{\beta}}\left\|\tilde{\boldsymbol{R}}^{-\top}\boldsymbol{e}_j\right\|_2 + \boldsymbol{e}_j^\top \tilde{\boldsymbol{R}}^{-1}\tilde{\boldsymbol{y}}$. It is observed that the quantities $\tilde{\beta}$, $\boldsymbol{R}^{-1}$, and $\tilde{\boldsymbol{y}}$ change between the parent node and the child node in a search tree. We can use two incremental updating strategy to efficiently modify the QR factorization and the matrix inverse to avoid recomputing these quantities from scratch.

Updating $\tilde{\beta}$ and $\tilde{\boldsymbol{y}}$. When a variable x_i is fixed, it is equivalent to deleting the i-th column in the matrix at the parent node. We can update the QR factorization using the standard technique [71] at the child node and compute the new $\tilde{\beta}$ and $\tilde{\boldsymbol{y}}$ in $O(n^2)$ time-complexity.

Updating $\tilde{\boldsymbol{R}}^{-1}$. Khan [84] proposed an efficient way to update the inverse of the matrix multiplication $(\tilde{\boldsymbol{R}}^\top\tilde{\boldsymbol{R}})^{-1}$ when a column is deleted (i.e., a variable is fixed, in our case) in $\tilde{\boldsymbol{R}}$ with $O(n^2)$ time-complexity. We can use this strategy to update $(\tilde{\boldsymbol{R}}^\top\tilde{\boldsymbol{R}})^{-1}$ at the child node. Then we can recover the quantities $\left\|\tilde{\boldsymbol{R}}^{-\top}\boldsymbol{e}_j\right\|_2 = \sqrt{\left\|\boldsymbol{e}_j^\top(\tilde{\boldsymbol{R}}^\top\tilde{\boldsymbol{R}})^{-1}\boldsymbol{e}_j\right\|_2}$ and $\tilde{\boldsymbol{R}}^{-1}\tilde{\boldsymbol{y}} = (\tilde{\boldsymbol{R}}^\top\tilde{\boldsymbol{R}})^{-1}\tilde{\boldsymbol{R}}^\top\tilde{\boldsymbol{y}}$.

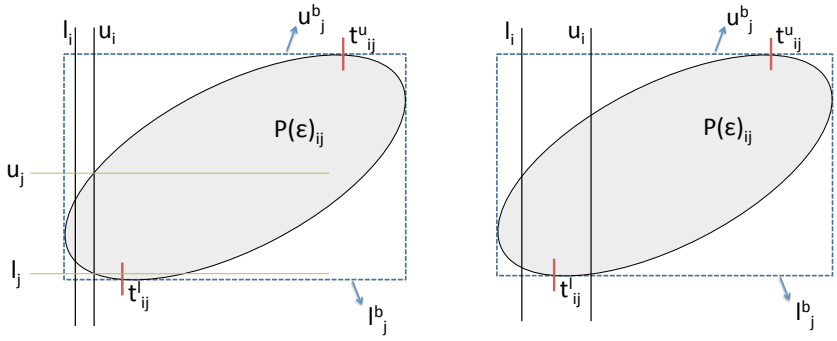
With the above updating mechanism, the complexity of computing the tangent box can be reduced to $O(n^2)$. We note that our current implementation involves copying data between the reversible data structure in CP Optimizer and the GNU Scientific Library (GSL) that we use for the linear algebra operations, which creates a significant overhead. Therefore while the theoretical results are established, we do not report our running times with this updating mechanism. A more efficient data structure that avoids copying between the two representations is required to efficiently implement this approach.

5.4.3 Approximate Bounds Consistency (ABC) Filtering

Before we introduce the next filtering algorithm, our notation is summarized as follows:

- $[l_j, u_j]$: The interval domain (local bounds) of variable x_j.
- $[l_j^b, u_j^b]$: The tangent box derived with Equation (5.9) and (5.10) of the ellipsoid $\mathcal{E}$ defined in Equation (5.2).
- v_j: A value that is within x_j's domain, i.e., $v_j \in I_D(x_j) = [l_j, u_j]$.

[2] Note that in our implementation, the inverse is computed by solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{I}$ for efficiency and reliability. However, this computation still has worst-case complexity of $O(n^3)$.



(a) The case that $l_i \leq u_i \leq t^l_{ij}$.

(b) The case that $l_i \leq t^l_{ij} \leq u_i$.

Figure 5.2: The 2D projection of the hyper-ellipsoid onto the $x_i x_j$ plane.

It is observed that if $\boldsymbol{l} \leq \boldsymbol{l}^b \leq \boldsymbol{u}^b \leq \boldsymbol{u}$, then the tangent box defines the bounds of the variables. However, a pair of bounds may lead to reductions in other variable domains. For example, in Fig. 5.2-a, the bounds l_i and u_i have the effect of increasing the lower bound l^b_j to l_j and decreasing the upper bound u^b_j to u_j.

To perform stronger domain reductions on the ellipsoid, first we need to determine the set of variables that can be used to infer reductions on the lower bounds or upper bounds of the variables. We explain the propagation algorithm below for the lower bound only since the propagation on the upper bound can be derived in a symmetric manner.

Proposition 1. *Let $P(\mathcal{E})_{ij}$ be the ellipse defined by the projection of the hyper-ellipsoid (Equation 5.2) onto the $x_i x_j$ plane. Then the variable x_i can be used to infer domain reductions on x_j's lower bound if and only if $l_i \leq u_i \leq t^l_{ij}$ or $t^l_{ij} \leq l_i \leq u_i$, where t^l_{ij} is the x_i value at the intersection of $x_j = l^b_j$ and the projected ellipse $P(\mathcal{E})_{ij}$.*

Proof. If $l_i \leq t^l_{ij} \leq u_i$ (Fig. 5.2-b), we can set $x_i = t^l_{ij}$, so that $x_j = l^b_j$, thus no domain reduction can be inferred to x_j's lower bound. In the other two cases where $l_i \leq u_i \leq t^l_{ij}$ (Fig. 5.2-a) or $t^l_{ij} \leq l_i \leq u_i$, since x_j is forced to take a value that is greater than l^b_j, we can increase x_j's lower bound. □

We refer to t^l_{ij} and t^u_{ij} as the touching points.

5.4.3.1 Computing the Touching Points

The touching points can be computed easily as a by-product of computing the axis-aligned tangent box (5.9) and (5.10). Since $\boldsymbol{p} = \sqrt{\beta}\boldsymbol{A}^{-\top}\boldsymbol{e}_j / \left\|\boldsymbol{A}^{-\top}\boldsymbol{e}_j\right\|_2$ uniquely defines the minimizer for the upper bound u^b_j, let $\boldsymbol{x}^*$ be the solution to the equation $\boldsymbol{p} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}$. We have:

$$(\boldsymbol{t}^u_j)^\top = [t^u_{1j}, \ldots, t^u_{j-1,j}, t^u_{j+1,j}, \ldots, t^u_{nj}]^\top = [x^*_1, \ldots, x^*_{j-1}, x^*_{j+1}, \ldots, x^*_n]^\top$$

Note that $\boldsymbol{t}^u_j$ is a $n-1$ dimensional vector and $x^*_j = u^b_j$. Using the symmetry property of the ellipsoid, $\boldsymbol{t}^l_j$ can be computed by reflecting t^u_j about the center of the ellipsoid.

The complexity of computing the touching points for all the variables, i.e., $\boldsymbol{t}^l_j, \boldsymbol{t}^u_j, \forall j$, is $O(n^3)$, as $\boldsymbol{x}^*$ can be computed in $O(n^2)$ for each variable, given that $\boldsymbol{A}^{-1}$ is known.

Proposition 2. *If x_i can be used to increase x_j's lower bound l_j according to Proposition 1, the value v_i^d that should be used to increase l_j is defined as*

$$v_i^d = \begin{cases} l_i, & \text{if } (t_{ij}^l - l_i)^2 \leq (t_{ij}^l - u_i)^2. \\ u_i, & \text{otherwise.} \end{cases}$$

Proof. Since $P(\mathcal{E})_{ij}$ is convex and x_j achieves its minimum l_j^b at $x_i = t_{ij}^l$, for any point x_j in $P(\mathcal{E})_{ij}$, we have x_j strictly larger than l_j^b if x_i takes any value other than t_{ij}^l. That is, x_j increases strictly when x_i moves away from t_{ij}^l. Therefore, the value (l_i or u_i) that achieves the minimum of the expression $\min((t_{ij}^l - u_i)^2, (t_{ij}^l - l_i)^2)$ determines x_j's lower bound. □

As Fig. 5.2-a depicts, we choose u_i in this example, as using l_i removes the valid value l_j.

Using Proposition 1 and 2, we can identify the set of variables and their values that can be used to increase the lower bound of a variable.

Definition 5.4.2. For each variable x_j, let S_j^l (S_j^u) be the set of all the variables that can be used to infer domain reductions on x_j's lower (upper) bound.

Definition 5.4.3. An assignment A: $x_i \mapsto I_D(x_i)$, $i \in S_j^l$ or S_j^u is said to be a determining assignment when $A(x_i) = v_i^d$.

When a variable takes a determining assignment, we can compute the reduced intersecting ellipsoid using the method in Section 5.4.2.1. The complete filtering algorithm for pruning the lower bound of a variable is presented in Algorithm 3. The upper bound pruning can be derived in a symmetric manner.

Algorithm 3 ABC filtering for x_j

Data: The local bounds: $l_1, \ldots, l_n$, $u_1, \ldots, u_n$, the tangent box for the ellipsoid $\mathcal{E}$: $l_1^b, \ldots, l_n^b$, $u_1^b, \ldots, u_n^b$, the touching points t_{ij}^l, β
Results: The filtered lower bound l_j'
Initialization: Set $l_j' = -\infty$, $\mathcal{F} = \{\}$
if $u_j < l_j^b$ **then**
  The constraint is not satisfiable
else
  Compute the set S_j^l and the associated assignments $A(x_i)$, $\forall i \in S_j^l$
  for each $i \in S_j^l$ **do**
    Let $\mathcal{F} = \{i\}$, compute the reduced intersecting ellipsoid $\mathcal{E}_F$
    Compute the tangent box $(l_j^F)^b$ of $\mathcal{E}_F$
    Set $l_j' = \max((l_j^F)^b, l_j')$
  end for
  if $u_i < l_j'$ **then**
    The constraint is not satisfiable
  else
    Set $l_j' = \max((l_j', l_j)$
  end if
end if

5.4.3.2 Complexity of the ABC Filtering Algorithm

The filtering algorithm reduces the domain of a single variable in $O(n^3)$ time-complexity. First, computing the tangent box takes $O(n^3)$ (or $O(n^2)$ if we use the updating strategy in Section 5.4.2.3). The

touching points require $O(n^2)$ as explained previously. In line 7, the sets S_j^l can be computed in $O(n)$. The for-loop at Line 8 requires $O(n^3)$, as Line 9 and 10 require $O(n^2)$ for computing the reduced ellipsoid (by updating the QR factorization) and the tangent plane for each of the i in S_j^l. Therefore the total time-complexity for filtering the domain for one variable is $O(n^3)$.

The complexity for filtering all the variables is therefore $O(n^4)$, which is the complexity when $S_j^l, \forall j$, contains $n-1$ variables. However, it is beneficial to have more variables in the set, as more and stronger pruning might be done.

5.4.4 Relative Strength of the Three Filtering Algorithms

It is clear that the ABC filtering algorithm is at least as strong as the BOX filtering algorithm, since ABC uses the tangent box as the starting point. The QCP filtering is at least as strong as ABC. ABC only considers the 2D projection of the hyper-ellipsoid onto each $x_i x_j$ plane, i.e., x_j is only tightened using the bounds of each x_i, independently. However, it is also possible to perform a higher dimensional projection of the hyper-ellipsoid and use the bounds of more than one variable together to do bound tightening. Consider the 3D projection of the hyper-ellipsoid and the reasoning among x_i, x_j, x_k. It is possible to use x_i and x_k, together, to tighten x_j, given that x_i and x_k intersects inside the hyper-ellipsoid. For this reason, ABC only achieves BC when $|S_j^l| = 1, |S_j^u| = 1, \forall j$, and the tangent box only achieves BC when $|S_j^l| = 0, |S_j^u| = 0, \forall j$.

5.4.5 Filtering Algorithm for the Axis-Aligned Ellipsoid (AAE)

We end this section with a note on a property of the ellipsoid constraint. Sections 5.4.1 to 5.4.3 define the filtering algorithms for problems with a general strictly convex ellipsoid structure. In particular, the filtering algorithm deals with a *rotated ellipsoid* structure, i.e., the off-diagonal entries of H in Equation (5.3) are not necessarily all equal to zero. For the *axis-aligned* ellipsoid, i.e., the matrix H is a diagonal matrix, bounds consistency can be achieved in a similar way as in the linear constraint using the lower and upper bounds of the variables. We describe an algorithm to solve the QCP formulation exactly for the axis-aligned ellipsoid by analyzing the feasible region defined by the quadratic constraint and the variable bounds as follows. First, we rewrite Equation (5.2) as $\sum_{i=1}^{n}(a_{ii}x_i - y_i)^2 \leq \beta$. Reorganizing the inequality with respect to x_j, we have

$$(a_{jj}x_j - y_j)^2 \leq \beta - \sum_{i=1, i\neq j}^{n} (a_{ii}x_i - y_i)^2. \tag{5.12}$$

The QCP formulation for the axis-aligned ellipsoid can therefore be rewritten as follows:

$$l_j^{BC} = \min x_j, \quad \text{subject to (5.12) and } \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}.$$

$$u_j^{BC} = \max x_j, \quad \text{subject to (5.12) and } \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}.$$

We show how to compute the lower bound l_j^{BC} first. Since we aim at minimizing x_j, the right hand side of Equation (5.12) should be maximized in order to achieve the largest quantity for $(a_{jj}x_j - y_j)^2$, or,

x_j. Therefore we have

$$(a_{jj}x_j - y_j)^2 \leq \beta - \min \sum_{i=1, i\neq j}^{n} (a_{ii}x_i - y_i)^2 = \beta - \sum_{i=1, i\neq j}^{n} \min (a_{ii}x_i - y_i)^2, \tag{5.13}$$

where

$$\min (a_{ii}x_i - y_i)^2 = \begin{cases} 0, & \text{if } l_i \leq \frac{y_i}{a_{ii}} \leq u_i. \\ (a_{ii}l_i - y_i)^2, & \text{if } \frac{y_i}{a_{ii}} < l_i. \\ (a_{ii}u_i - y_i)^2, & \text{if } u_i < \frac{y_i}{a_{ii}}. \end{cases}$$

Note that the equality holds in Equation (5.13) because all the x_i terms in the summation are completely independent of each other. It follows that

$$l_j^{BC} = \frac{-\sqrt{\beta - \sum_{i=1, i\neq j}^{n} \min(a_{ii}x_i - y_i)^2}}{a_{jj}} + \frac{y_j}{a_{jj}}.$$

From the symmetry property of an ellipsoid, we have

$$u_j^{BC} = \frac{\sqrt{\beta - \sum_{i=1, i\neq j}^{n} \min(a_{ii}x_i - y_i)^2}}{a_{jj}} + \frac{y_j}{a_{jj}}.$$

Note that if the quantity in the square root expression is less than zero, i.e., $\beta - \sum_{i=1, i\neq j}^{n} \min(a_{ii}x_i - y_i)^2 < 0$, then the constraint is immediately unsatisfiable.

The complexity of the algorithm is $O(n^2)$ for filtering all the variables. In our implementation, we can detect such axis-aligned ellipsoids and apply this algorithm instead of the ones for rotated ellipsoids.

5.5 Branching Rules

We propose variable and value ordering rules inspired by the search strategy of DEBS. Recall that the static variable ordering in DEBS tries to minimize the branching factor at the top of the tree so that it is easier to find feasible solutions. In CP, this is the same as statically choosing a variable with minimum domain size. We therefore use the standard dynamic variable selection rule that chooses a variable with the smallest domain, as dynamic variable ordering heuristics are generally superior to their static counterparts [133]. The value ordering rule in DEBS always assigns a variable to the integer value closest to the center of the ellipsoid of the objective function so that the search greedily chooses the best integer value at the current node with the hope of finding good feasible solution quickly. In our implementation, suppose x_j is the variable chosen for branching, we first compute the center of the ellipsoid c_j in the j-th dimension given the reduced ellipsoid, and then round it to the nearest integer. When the search backtracks, we assign x_j to the next nearest integer to c_j, and so on.

5.6 Experimental Results

Experimental Setup. We design two experiments. The goal of the first experiment is to evaluate the impact of our filtering algorithms and the branching heuristics compared to a default CP model. The second experiment compares our new CP approach to the best known exact approaches for IQCPs.

For the first experiment, we use IBM ILOG CP Optimizer v12.6.3 with its default settings. The four filtering algorithms, i.e., BOX, ABC, QCP, and AAE are implemented in CP Optimizer as customized global constraints. The branching rules are implemented as customized variable and value choosers (denoted with the symbol "+b" in our results). We use CPLEX v12.6.3 for solving the QCPs. We report the arithmetic mean CPU time "time" in seconds, and the arithmetic mean number of choice points "chpts" to find and prove optimality for each problem set.

For the second experiment, we use CPLEX v12.6.3,[3] BARON v16.4.7 (using CPLEX v12.6.3 as its LP/MIP solver) and the SDP solver BiqCrunch downloaded from the website [88] for comparison. Since there are four versions of the SDP solver that deal with problem-specific structures, the SDP results presented are the best version for each individual problem *instance*, representing the "virtual best" SDP solver. All solvers are executed with their default settings. The DEBS algorithm is written in C.

The CPU time limit for each run on each problem instance is 3600 seconds. All experiments were performed on a Intel(R) Xeon(R) CPU E5-1650 v2 3.50GHz machine (in 64 bit mode) with 16GB memory running MAC OS X 10.9.2 with one thread.

5.6.1 Problem Sets

We experiment on medium size problems in five problem classes. The problem size of each set is chosen with the aim for a mix of solvable and non-solvable instances across the default CP Optimizer and the three filtering algorithms.

Binary Quadratic Programming (BQP) Problem. The BQP problem is defined as:

$$\min_{\boldsymbol{x}\in\{0,1\}} \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x},$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{f} \in \mathbb{R}^n$. BQPs arise in many combinatorial optimization problems such as task allocation [100], quadratic assignment [57], and max-cut problems [87]. We experiment on the Carter type problems [40] divided into four sub-sets of instances with size 40 ($p = 0.2$), 40 ($p = 0.3$), 50 ($p = 0.2$) and 50 ($p = 0.3$), respectively, where p is a problem generation parameter that controls the magnitude of the diagonal entries.

Exact Quadratic Knapsack Problem (EQKP). The EQKP [99] is defined as:

$$\min_{\boldsymbol{x}\in\mathcal{C}} \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^\top \boldsymbol{x}, \quad \mathcal{C} = \{\boldsymbol{x} \in \{0,1\} : \boldsymbol{c}_1^\top \boldsymbol{x} = K,\ \boldsymbol{c}_2^\top \boldsymbol{x} \le B\},$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{f} \in \mathbb{R}^n$, $\boldsymbol{c}_1 \in \mathbb{R}^n$ is a vector equal to ones, $\boldsymbol{c}_2 \in \mathbb{R}^n_+$, $K \in \mathbb{Z}_+$, $B \in \mathbb{R}^+$. The objective is to minimize a quadratic function subject to a cardinality constraint and a knapsack constraint. The EQKP is a extension of the BQP, maximum diversity problem [106], quadratic knapsack problem [39], and exact linear knapsack problem [38]. EQKPs arise in a wide range of real world applications such as wind farm optimization [147, 163]. We experiment on the EQKP and a variation in Section 3.3 with binary domains $x_j \in \{0, 1\}$ relaxed to $x_j \in \{0, 1, 2\}$. We use three sub-sets of the instances with size 10, 20 and 30.

[3] A major improvement was made in solving IQCPs in CPLEX v12.6.3 [31].

Box-constrained ILS (BILS) Problem. The BILS problem can be defined as:

$$\min_{\boldsymbol{x}\in\mathcal{C}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{C}=\{\boldsymbol{x}\in\mathbb{Z}^n : \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u},\ \boldsymbol{l}\in\mathbb{Z}^n,\ \boldsymbol{u}\in\mathbb{Z}^n\}.$$

The problem minimizes a least squares expression subject to general integer bounds. Such problems exist in elevator scheduling [94] and signal processing [43]. We generate problems the same way as Chang et al. [43], with medium size variable domains ($0 \leq x_i \leq 10, \forall i$) and medium level of noise ($\sigma = 0.05$). We use five sub-sets of the instances with size 10, 20, 30, 40 and 50.

Box-constrained and Ellipsoid-constrained ILS (BEILS) Problem. The BEILS problem can be defined as:

$$\min_{\boldsymbol{x}\in\mathcal{C}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{C}=\{\boldsymbol{x}\in\mathbb{Z}^n : \|\boldsymbol{A}\boldsymbol{x}\|_2^2\leq\alpha, \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u},\ \boldsymbol{l}\in\mathbb{Z}^n,\ \boldsymbol{u}\in\mathbb{Z}^n\},$$

where α is a constant. In addition to the least squares objective function, the BEILS problem is also subject to a least squares constraint. Such problem can exist in signal processing [48]. We generate problems the same way as those in Chang et al. [42] on the ellipsoid-constrained ILS problem then add medium size variable domains ($-10 \leq x_i \leq 10, \forall i$). We use five sub-sets of the instances with size 10, 20, 30, 40 and 50.

Quadratic Lateness Scheduling Problem (QLSP). The QLSP can be defined as:

$$\min \sum_{j=1}^{n} (S_j + p_j - d_j)^2, \quad \text{s.t. } \texttt{disjunctive}(\{S_1, ..., S_n\}, \{p_1, ..., p_n\}),\ S_j \geq 0, \forall j,$$

where S_j, p_j, and d_j are the start time, processing time, and due date of job j. The QLSP is a single machine scheduling problem with the goal of minimizing the sum of the quadratic lateness of the jobs. We generate problems the same way as those in Schaller [135]. We use four sub-sets of the instances with size 5, 10, 15 and 20.

Each problem set includes 10 instances of each size, e.g., the BQP problems includes 4 sub-sets of size 10 for 40 instances.

5.6.2 Results of Experiment 1

We present the results of Experiment 1 in Tables 5.1 and 5.2. From Table 5.1, it is clear that the ELLIPSOID constraint significantly improves the performance of the default CP Optimizer for the BQPs, the BILS problems, and the BEILS problems both in terms of running time and number of choice points. Without the reasoning from the ELLIPSOID constraint, the default CP Optimizer cannot prove optimality for any instances of these three problem types. For the EQKPs, the ELLIPSOID constraint (BOX) is able to decrease the number of choice points by a factor of 1.5 for the {0,1} problems and almost a factor of 2 for the {0,1,2} problems. But the extra computation makes the running time worse than that of the default CP Optimizer.

For QLSPs, CPO achieves the best average running time. Interestingly, for the instances where all the filtering algorithms are able to prove optimality, the number of choice points is the same for

	CPO		BOX		ABC		QCP		AAE	
Problem	time	chpts	time	chpts	time	chpts	time	chpts	time	chpts
BQP	-	-	**11.12**	**17169**	1204.89^{85}	10565	-	-	N/A	N/A
BILS	-	-	**44.61**	**61552**	871.80^{84}	19229	2810.01^{40}	3836	N/A	N/A
BEILS	-	-	**362.14**^{94}	**167162**	1480.68^{64}	10301	3092.14^{38}	1637	N/A	N/A
EQKP{0,1}	**23.48**	667740	44.09	**426255**	745.79^{87}	60789	2494.77^{67}	2494	N/A	N/A
EQKP{0,1,2}	**83.98**	1803982	99.70	**932035**	481.44^{90}	63177	2256.95^{57}	2273	N/A	N/A
QLSP	**136.34**	**962400**	166.16	**962400**	754.07^{85}	438394	2432.98^{50}	8631	139.09	**962400**

Table 5.1: A comparison of default CP Optimizer and the four filtering algorithms. The running times are reported in seconds. Bold numbers indicate the best approach for a given problem set. The symbol '-' means that no problem instances were solved to optimality within 3600 seconds. The superscripts indicate the percentage of instances solved to optimality within 3600 seconds. If no superscript is indicated, all of the instances are solved. The symbol 'N/A' indicates that the problem cannot be solved with the filtering algorithm.

	CPO+b		BOX+b		ABC+b		QCP+b		AAE+b	
Problem	time	chpts	time	chpts	time	chpts	time	chpts	time	chpts
BQP	-	-	**8.28**	**12611**	1099.82^{92}	12575	3595.85^{3}	1069	N/A	N/A
BILS	-	-	**0.06**	225	5.21	228	314.16	**221**	N/A	N/A
BEILS	-	-	**0.47**	426	213.47	**394**	1713.89^{82}	360	N/A	N/A
EQKP{0,1}	**16.40**	247896	24.84	**193269**	578.74^{90}	72134	1868.71^{57}	2550	N/A	N/A
EQKP{0,1,2}	50.04	521037	**46.77**	**269937**	407.59^{90}	39076	2304.01^{60}	2365	N/A	N/A
QLSP	**20.63**	**347665**	31.47	**347665**	602.77^{90}	265252	2249.57^{50}	8516	22.29	**347665**

Table 5.2: A comparison of default CP Optimizer and the three filtering algorithms with the branching rules. All notations are the same as in Table 1.

all the filtering algorithms. Our preliminary investigation shows that the `disjunctive` constraint has a significant impact on reducing variable domains for the QLSP, therefore determining the number of choice points regardless of the strength of propagation of the ellipsoid constraint. It is worth pointing out that the convex ellipsoid structure of the QLSP has a simple axis-aligned structure. We report the running time of the three filtering algorithms for rotated ellipsoids, nonetheless, to show the additional running time incurred. In practice, we can automatically identify the axis-aligned property and avoid applying these inference algorithms for this type of problem. In addition, our investigation shows that the default CPO propagation does not achieve BC for axis-aligned ellipsoids.

Among the three filtering algorithms for rotated ellipsoids, BOX performs the best in terms of running time to prove optimality. ABC and QCP both find better primal solutions in fewer nodes but BOX finds better solutions in less time. On problems where all three algorithms are able to prove optimality, the tree size of BOX is about 15% larger than that of ABC and QCP. This result is somewhat surprising and suggests that variable fixing leads to strong inference. We observe that the reduced ellipsoid obtained after fixing a variable often has a much tighter tangent box on the unfixed variables compared to that of the original ellipsoid. However, we also observe that ABC can sometimes achieve one order of magnitude improvement in tree size compared to BOX on some larger instances. We would like to investigate the problem characteristics that result in such difference. In addition, it is possible to improve the time complexity of ABC to $O(n^3)$ by heuristically choosing only one variable that leads to the most domain

reduction from the set S_j^l (S_j^u). In the future, we would like to investigate such heuristics that achieve a good balance between filtering power and time complexity. Note that the lower number of choice points for ABC and QCP are misleading because neither prove optimality for all instances within the time limit. However they are good indicators on the number of nodes that can be visited when these two algorithms are applied.

From Table 5.2, it is observed that the new branching rules greatly improve the number of choice points, the running time, and the percentage of optimal solutions found for most approaches. However CPO still cannot prove optimality for the BQPs, the BILS problems, and the BEILS problems. The most significant reduction is observed on the BILS problem and the BEILS problem, followed by the QLSP, where the variable domains are much larger than the other two types of problems: a good branching strategy apparently is especially important for problems with large domains. It is particularly striking to see that the BILS problems are solved at least five orders of magnitude faster by CP through the use of the ellipsoid constraint and the branching rule.

5.6.3 Results of Experiment 2

In this section, we compare our best CP results (using the BOX filtering for rotated ellipsoids and the AAE for axis-aligned ellipsoids with the branching rules) with the best known exact approaches. From Table 5.3, it is observed that our new CP approach significantly outperforms the general MINLP solver BARON and it is competitive with state-of-the-art MIP solver CPLEX, running at the same order of magnitude as CPLEX for BQPs and BILS problems. While our CP approach is one order of magnitude slower than CPLEX for EQKPs, it is almost two orders of magnitude faster for BEILS problems and it is significantly better for QLSPs. The reason that CPLEX performs particularly poorly on QLSPs is that the modeling of the disjunctive relationship among the jobs involve big-M constraints, which result in weak dual bounds. As future work, we would like to apply our CP approach to more complicated scheduling problems, where quadratic component is only one of many components.

	CPO	BEST CP	CPLEX	BARON	DEBS	SDP
Problem	time	time	time	time	time	time
BQP	-	8.28	37.06	38.03	**0.24**	0.69
BILS	-	0.06	0.02	2715.04^{26}	**0.01**	N/A
BEILS	-	**0.47**	14.79	3248.25^{16}	N/A	N/A
EQKP{0,1}	23.48	24.84	4.49	6.23	**0.24**	114.01
EQKP{0,1,2}	83.98	46.77	4.27	429.62^{93}	**0.34**	3593.55^{2}
QLSP	136.34	**22.29**	1588.15^{65}	1855.02^{50}	N/A	N/A

Table 5.3: A comparison of default CP Optimizer and our best setting: BOX+b for rotated ellipsoids and AAE+b for axis-aligned ellipsoids. All notations are the same as in Table 1. The symbol "N/A" indicates that the problem cannot be solved with the approach.

The SDP approach, while being the state of the art for the BQPs, is limited to problems with binary domains. For example, it cannot be used to solve the $\{0, 1, 2\}$ EQKPs without transforming the problem back to binary domains at the cost of introducing additional variables. Nevertheless, we perform this transformation using the standard technique by representing the integer variables with their binary expansion [55]. Results show that the SDP approach does not solve the relaxed EQKPs efficiently due

to the very slow improvement of the dual bound. Similarly, DEBS cannot be applied to all the problem classes due to its specialized nature. In contrast, BARON is the most general solver tested here (along with CPO) and these results on strictly convex IQCPs do not reflect its more general problem solving power [145].

Parameter Tuning for CPLEX. As CPLEX clearly outperforms the other general solvers in our experiment, we aim at achieving the best performance of CPLEX and compare to our best CP setting. Specifically, we investigate a crucial aspect of modern MIP solvers: parameter tuning. Parameter tuning targets finding the best set of parameters for a specific class of problems and therefore may drastically improve performance.

We use CPLEX's parameter tuning tool as follows. First, we classified the instances into five categories based on problem types, i.e., one category for each problem type. We assume that the problem characteristic are different between these problem types therefore avoiding potential performance loss when using the tuning tool. We set a total tuning time of two days, with 3600 seconds per run. All the training instances are also used as the testing instances. This approach obviously biases the results in favour of the tuned parameters, however, we did this as we were seeking to evaluate the maximum improvement we could expect from tuning.

In additional to using CPLEX's tuning tool, we also experiment with two different node processing strategies. CPLEX can be set to solve a regular linear relaxation or a quadratic relaxation at each node for IQCPs. Since neither of these two strategies dominates each other in general [47], we experiment with both. All the rest of the parameters are kept default.

	Default	Linear	Quadratic	Tuned	Virtual Best	BEST CP
Problem	time	time	time	time	time	time
BQP	37.06	37.66	37.65	37.06	33.23	**8.28**
BILS	0.02	**0.01**	**0.01**	**0.01**	**0.01**	0.06
BEILS	14.79	85.96	10.48	9.90	4.26	**0.47**
EQKP{0,1}	4.49	4.52	4.52	2.70	**2.62**	24.84
EQKP{0,1,2}	4.27	4.16	4.16	5.74	**4.13**	46.77
QLSP	1588.15^{65}	1648.87^{65}	1652.03^{63}	1588.15^{65}	1582.20^{68}	**22.29**

Table 5.4: A comparison of five different CPLEX settings and our best CP setting. The notations "Linear" and "Quadratic" denote the two different node processing strategies. The notation "Virtual Best" is the best version of CPLEX for each individual problem instance.

Table 5.4 shows that CPLEX's tuning tool reduces the mean running time for the BEILS problems by a factor of 1.5 and the {0,1} EQKPs by a factor of 1.7, but increases the running time for the {0,1,2} problems by a factor of 1.3. For the BQPs and the QLSPs, the tuning tool does not find parameter settings different from the default so the performance remains unchanged. We conclude that though parameter tuning is an effective technique, our results indicate that it is unlikely to make orders of magnitude improvement or to make a significant portion of the non-solvable problems solvable.

Comparing the two node processing strategies, results show that solving the quadratic relaxation yields much better performance than the linear relaxation for the BEILS problems, but slightly underperforms on the QLSPs. It is worth noting that the default setting is able to make good decision on dynamically choosing the type of relaxation.

Finally, we look at the performance of the "virtual best" CPLEX solver, which is the best setting of CPLEX for each individual problem instance. The virtual best solver shows a more significant improvement on the BEILS problems, reducing the running time by a factor of 3.5 compared to the default setting. However, when compared to our best CP setting, the relative performance remains the same: our CP approach performs better on the BQPs, BEILS problems, and QLSPs.

5.7 Conclusion

We propose a CP-based approach to solve strictly convex IQCPs via a novel ELLIPSOID constraint with three different filtering algorithms and variable/value ordering heuristics. The constraint and branching heuristics are based on the geometry of the strictly convex quadratic function. We experiment with a variety of problems and show significant improvement over the default CP Optimizer and competitive results to general state-of-the-art solvers CPLEX and BARON.

For future work, it is interesting to experiment with our filtering algorithms on other problem types. Although ABC and QCP perform worse than BOX for the problem types tested here, it is possible that ABC and QCP can perform better for problems where variable fixings do not take place frequently. For the same reason it is also interesting to implement the filtering algorithms in a MIP-based solver such as SCIP [155] where branching is typically done by partitioning variable domains instead of fixing variables. In general, our technique can be integrated with other solvers provided they represent bounds on variables and have an"inference loop". It may also be interesting to design an adaptive strategy to select different filtering algorithms at different levels of the tree, e.g., apply QCP and ABC near the top of the tree and apply BOX elsewhere.

As for implementation, though we have established the theoretical foundation of the updating strategy, our current implementation creates a significant overhead because of the data copying between the reversible data structure in CP Optimizer and the GNU Scientific Library (GSL) that we use for the linear algebra operations. One possible solution is to implement the linear algebra operations using the reversible data structure, thus avoiding the memory copying. In addition, currently we use matrices to store the information used for updating. Ideally, we would like to design more efficient data structure that only keeps track of the minimum amount of information required for the updating mechanism.

More broadly, we propose that it may be possible to develop CP as a basis for MINLP. MINLPs are challenging optimization problems that arise in many industrial applications. As CP is not dependent on a strong linear relaxation to bound the search, it may be valuable to study inferences that can be made for common non-linear constraints as we have done here. We hope that this work might serve as a step in this direction.

Chapter 6

Conclusion

In this chapter, we provide a summary and restate the contributions of this dissertation. We conclude with some potential future research topics.

6.1 Summary

In this dissertation we study exact methods for solving the strictly convex integer quadratically constrained problem (IQCP), an important class of optimization problems that has numerous applications in industry and theoretical science. We exploit the similarities among three tree search algorithms, i.e., discrete ellipsoid-based search (DEBS), mixed integer programming (MIP), and constraint programming (CP) and integrate their techniques to develop efficient hybrid algorithms for solving IQCPs. We demonstrate the state-of-the-art performance of our hybrid algorithms by comparing them to the best known exact methods in the literature. In the following paragraphs, we summarize our investigation on different approaches of hybridizing the techniques from DEBS, MIP and CP.

In Chapter 1, we introduce the IQCP and provide an overview on the exact methods for solving the IQCP. We describe the current status of DEBS, MIP and CP and identify the strengths and weaknesses of each approach, thus motivating the importance of developing efficient hybrid algorithms. We then provide an outline of our research and describe the main contributions of this dissertation.

In Chapter 2, we provide the necessary notations and key concepts needed in this dissertation. We formally define the IQCP and introduce the mutual reformulation of the IQCP and the integer least squares (ILS) problem, demonstrating the theoretical foundation of our hybrid algorithms. We then review the solving techniques of MIP, CP, and DEBS for IQCPs, which is, to the best of our knowledge, the first time in the literature that anyone has taken a broader view of the problem that is often studied independently by different research fields. We end this chapter with an overview of how the existing literature relates to our new approaches for solving IQCPs.

In Chapter 3, we first show how DEBS can be incorporated into a MIP solver as a component enhancement to strengthen three key aspects of modern MIP solvers: presolving, cutting planes, and primal heuristics. Our hybrid algorithm achieves the state of the art for the binary quadratic programming (BQP) problem, an important class of problems in OR, and three types of the ILS problems that frequently arise in communications applications. A major advantage of our hybrid algorithm is that, unlike the specialized BQP solvers that can only solve problems with binary variables, our hybrid algorithm

can be applied to a much larger class of problems that involve general integer variables.

We then show how DEBS can be incorporated in CP as a node processing enhancement to strengthen the filtering power of CP for IQCPs. Since DEBS can be regarded as a specialized CP search with its own branching heuristics and propagation rules tailored for strictly convex quadratic functions, we use this insight to integrate linear constraints into DEBS so that the domains of the variables are further reduced by the intervals implied by the linear constraints. This work is the first that generalizes DEBS to incorporate general linear constraints, greatly broadening the class of problems that DEBS can solve. As test set, we experiment with the exact quadratic knapsack problem (EQKP) and a variation and demonstrate state-of-the-art performance.

The hybridization in Chapter 4 is done by integrating DEBS, MIP and CP together in the constraint integer programming (CIP) paradigm. We augment the SPREAD global constraint with techniques beyond its traditional filtering algorithm, i.e., novel relaxations and cutting planes that are derived based on the combinatorial structure of the SPREAD constraint and the geometric reasoning from DEBS. We apply our CIP approach to the variance minimization problem with the application to two load balancing problems: the load balancing nurse-to-patient assignment problem (NPAP) and the balanced academic curriculum problem (BACP). Together with the augmented GCC constraint and problem specific branching heuristics, our CIP model outperforms both our new mixed integer quadratic programming (MIQP) model in CPLEX and the previous best known CP model for the NPAP. For the BACP, both our new CIP and MIQP models outperform the previous best known CP model.

Building upon the previous chapters, in Chapter 5 we develop a novel CP approach to solve general IQCPs, allowing any side constraints that fit in the CP framework. We propose a novel global constraint, the ELLIPSOID constraint that propagates based on strictly convex quadratic constraints. We formally define the classical bounds consistency notation for the ELLIPSOID constraint and propose three filtering algorithms, each with different filtering power and time complexity. Despite the natural limitation of integer domains in CP, our filtering algorithms can be applied to variables with real domains, thus broadening its application to approaches that support real variables, such as MIP solvers. We implement the filtering algorithms and the accompanying branching heuristics in IBM ILOG CP Optimizer and experiment with five sets of benchmark instances. Our results show orders of magnitude improvement over the default CP Optimizer. When compared with the best known algorithms in the literature, our results demonstrate that the new CP approach is competitive and achieves a new state of the art for some problem domains.

The central thesis of this dissertation is that integrating techniques from DEBS, MIP and CP results in efficient hybrid algorithms that perform better than using each method individually for solving IQCPs. Abundant geometric information can be extracted from DEBS and formulated as inference techniques that can be used in any tree search algorithm to prune search space.

6.2 Contributions

The main contributions of this dissertation are summarized as follows:

- **State-of-the-Art Hybrid Algorithms:**
  - **Strictly Convex Quadratically Constrained Problems with Variable Bounds:** We propose a novel hybrid algorithm that integrates MIP and DEBS to solve the class of problems

with strictly convex quadratic constraints and variable bounds. The key idea is to incorporate DEBS-based presolving, cutting planes, and a primal heuristic to enhance MIP solvers. We demonstrate the performance of our hybrid algorithms on the binary quadratic programming problems and the integer least squares problems, which have significant influence in the research fields of OR and communications, respectively.

  - **Strictly Convex Quadratically Constrained Problems with Linear Constraints:** We propose a novel CP and DEBS hybrid algorithm to solve strictly convex quadratically constrained problems with linear constraints. The key idea is to integrate linear constraints into DEBS so that the domains of the variables are not only reduced by the the bounding ellipsoid but also by the linear constraints. Results on the test set, the exact quadratic knapsack problem and a variation, show state-of-the-art performance.
  - **Variance Minimization Problems:** We propose a three-way hybrid algorithm that combines MIP, CP and DEBS for variance minimization problems and implement our algorithm in the constraint integer programming solver, SCIP. The key idea is to augment the traditional SPREAD global constraint with MIP's relaxation and cutting planes techniques, which are derived from the combinatorial structure of the SPREAD constraint and the geometric reasoning from DEBS. Results on two load balancing problems show that our CIP approach outperforms the best known CP approach in the literature.

- **A New Global Constraint for Strictly Convex IQCPs:** We propose a novel global constraint, the ELLIPSOID constraint, that filters w.r.t. strictly convex quadratic constraints. Our novel CP approach that combines the ELLIPSOID constraint and a pair of variable/value ordering heuristics brings CP within an order of magnitude of the state-of-the-art techniques for the IQCPs tested. Our CP approach also achieves the state of the art when compared with non-problem specific solvers such as CPLEX on some problem domains, showing the potential of developing CP as a basis for mixed integer nonlinear programming (MINLP) problems.

- **Bridging Different Research Fields for Solving IQCPs:** For the first time in the literature, we bring together the research of the OR, CP, and communications communities. We conduct extensive experiments that compare their traditional approaches and our novel hybrid algorithms, identifying key characteristics impacting the performance of each approach. We believe such a connection between different research fields can provide new insights into IQCPs and may inspire future innovation.

6.3 Future Work

As more immediate future work has been presented in their respective chapters (e.g., Sections 3.4, 4.8 and 5.7), here, we take a broader view and propose several lines of research that could be further explored.

6.3.1 CP as a Basis for MINLP

We have developed state-of-the-art hybrid algorithms for solving strictly convex IQCPs in this dissertation, nevertheless, our ultimate goal is to solve an even larger class of problems, beyond IQCPs. We propose that it may be possible to develop CP as a basis for mixed integer nonlinear programming

(MINLP). MINLPs are challenging optimization problems that arise in many industrial applications. The greatest challenges for solving MINLPs efficiently come from the higher order nonlinearity and the nonconvexity. Current MINLP solvers are based on mixed integer linear programming techniques and their performance depends heavily on the strength of the outer-approximated linear relaxations of the nonlinear functions. As CP is not dependent on a strong linear relaxation to bound the search, it may be valuable to investigate inferences that can be made for common nonlinear constraints, similar to what we have done for the strictly convex quadratic constraint.

In order to generalize our CP approach from the pure integer case to the *mixed* integer case, the filtering algorithms need to make correct inference both on integer and continuous variables so that they can be executed at each node of the search tree. One interesting question arises as to when the relaxed problems should be solved. In the MIP literature, a relaxed problem is solved at each node of the tree, however, it may be advantageous to delay solving the relaxed problems until after all the integer variables have been instantiated. As shown in some communications applications [44], a potential benefit of this approach is that a majority of the sub-problems might be pruned early in the search due to constraint propagation, thus avoiding solving a large number of the time-consuming relaxed problems. It may therefore also be of interest to develop branching heuristics that lead to fewer relaxed problems that need to be solved.

One of the barriers for MIP, and possibly for CP, to solve general MINLPs is nonconvexity. In CP, bounds consistency implicitly works with convex interval domains. A nonconvex constraint may results in many domain holes so that maintaining bounds consistency might not achieve meaningful filtering. To overcome the nonconvexity, it may be possible introduce the convex reformulation techniques [26] from the OR literature so that it may be easier to compute the rigorous enclosure of a nonlinear constraint. In the quadratic case, for example, convexity theory [34] states that the matrix that describes the quadratic terms of a strictly convex quadratic function is positive definite, meaning that it has only positive eigenvalues. Therefore, if we compute the eigenvalues and fix the variables that correspond to the non-positive eigenvalues, the resulting sub-problem consists of only strictly convex quadratic functions, which can then be filtered with our ellipsoid constraint. As an important application, we would like to investigate problems with the second-order cone constraint [103], which has only one negative eigenvalue. The problem becomes a strictly IQCP after fixing a single variable that corresponds to the negative eigenvalue. In addition, new enumeration techniques such as spatial branching [23] in the OR literature could be introduced to CP to efficiently handle the branching of the continuous variables in nonconvex constraints. It may also be possible to bound a nonlinear constraint with a known structure where the filtering algorithms are already available, e.g. using a minimum volume ellipsoid that encloses some nonlinear constraint.

Another important aspect is the extension of the classical consistency notions in CP for more appropriately defining the inference power of the filtering algorithms for nonlinear constraints. In the OR literature, the filtering algorithms in MINLP solvers are usually not executed until they reach a certain level of consistency [126], since it may be excessively computationally expensive. In accordance to what we have shown in Chapter 5, achieving bounds consistency for a strictly convex quadratic constraint is already difficult without using the time-consuming optimization-based bound tightening techniques. Therefore, it is also important to develop more light-weight filtering algorithms that achieve a balance between filtering power and time-complexity.

6.3.2 New Mechanisms in CP

Hybrid Propagation and Branching Mechanisms. We propose that it may be possible to design efficient hybrid propagation mechanism in CP. Inspired by DEBS, which only filters the variable that is being considered at the current decision level, we can apply filtering algorithms selectively to only the variable that has been selected but not assigned a value yet. For the ELLIPSOID constraint, for example, filtering selectively reduces the time complexity by one order of magnitude (e.g., the BOX filtering is $O(n^2)$ for a single variable, as opposed to $O(n^3)$ for all the variables). A potential advantage of filtering selectively is that, if the domain reductions of the variables other than the one to be instantiated are not contributing much to a domain wipe-out, substantial time can be saved from filtering these variables, while preserving most of the filtering power. In the future, we would also like to generalize the idea of selective filtering to some common global constraints. We believe that a hybrid propagation mechanism that combines the traditional propagation mechanism and the selective filtering is a promising direction for achieving a better performance.

It is also interesting to design hybrid branching mechanisms in CP, e.g., instead of fixing variables, we can branch in a similar way as MIP by partitioning variable domains, which may lead to better performance for some filtering algorithms. For example, as shown in Chapter 5, although ABC and QCP perform worse than BOX for the problem types tested in this dissertation, we suspect that ABC and QCP can perform better when branching by partitioning. Therefore, we propose that it may also be possible to design an adaptive branching strategy to select different filtering algorithms and different branching mechanism, at different depths of the tree, e.g., apply the more expensive but potentially more powerful filtering algorithms near the top of the tree and apply lighter weight filtering algorithms elsewhere.

For both hybrid mechanisms, efficient heuristics and machine learning techniques could be explored to adaptively select the best branching and propagation mechanisms on the fly.

Global Inference and Dual Bounding Mechanisms. One of the main challenges for solving MINLPs with CP is the lack of a global view on the entire problem [150], such as the dual bounding mechanism in MIP. We propose that it may be possible to use the multi-valued decision diagram (MDD) [9] to make stronger inference from a global perspective and as a general dual bounding mechanism for MINLPs. A MDD is a graphical data structure originally introduced to compactly represent Boolean functions [8]. More recently, MDDs have been applied to enhance propagation and provide an optimization based bound in CP [19]. We believe that it is a promising direction to represent the system of nonlinear constraints as MDDs to obtain tighter bounds on the objective function. We would also like to investigate MDD-based filtering algorithms for common nonlinear constraints. Finally, as MDDs have only been known to be applied to discrete optimization, it is also of great interest to formally extend the MDD for problems with both integer and continuous variables.

Appendices

Appendix A

Additional Results of Chapter 3

A.1 Complete Results of the ILS Problems

$c = 3$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.01	0.07	0.02	0.01	0.01	0.01	0.01	0.01	0.01
20	0.01	0.01	0.13	0.01	0.01	0.01	0.01	0.01	0.01
30	0.01	0.01	0.17	0.01	0.01	0.01	0.01	0.01	0.03
40	0.02	0.03	2.22	0.01	0.01	0.01	0.01	0.01	0.02
50	0.02	0.02	0.98	0.01	0.01	0.52	0.01	0.01	0.77
60	0.03	0.04	6.49	0.01	0.01	7.52	0.01	0.01	2.12
70	0.04	0.06	6.08	0.01	0.01	119.85	0.01	0.02	1.32
80	0.06	0.11	15.93	0.01	0.01	206.00	0.01	0.02	1.71
90	0.07	0.09	11.76	0.01	0.01	83.71	0.01	0.01	0.21
100	0.07	0.12	190.05	0.01	0.01	147.26	0.01	0.01	0.43

$c = 10$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
20	0.01	0.01	0.11	0.01	0.01	0.01	0.01	0.01	0.01
30	0.01	0.01	0.11	0.01	0.01	0.01	0.01	0.01	0.02
40	0.02	0.01	0.78	0.01	0.01	0.01	0.01	0.01	0.03
50	0.02	0.02	1.30	0.01	0.01	0.21	0.01	0.01	0.60
60	0.03	0.04	4.96	0.01	0.01	7.63	0.01	0.01	1.08
70	0.04	0.05	4.39	0.01	0.01	33.80	0.01	0.02	0.82
80	0.06	0.07	37.74	0.01	0.01	228.31	0.01	0.02	0.51
90	0.07	0.09	56.76	0.01	0.01	216.00	0.01	0.02	0.46
100	0.09	0.09	38.94	0.01	0.01	72.01	0.01	0.02	0.06

$c = 100$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01
20	0.01	0.01	0.19	0.01	0.01	0.01	0.01	0.01	0.01
30	0.01	0.01	0.27	0.01	0.01	0.01	0.01	0.01	0.01
40	0.02	0.03	0.62	0.01	0.01	0.01	0.01	0.01	0.04
50	0.03	0.03	3.27	0.01	0.01	0.21	0.01	0.01	0.59
60	0.02	0.04	1.39	0.01	0.01	7.63	0.01	0.01	2.50
70	0.04	0.09	5.23	0.01	0.01	33.80	0.01	0.01	0.89
80	0.05	0.09	7.57	0.01	0.01	128.31	0.01	0.01	2.64
90	0.06	0.10	12.08	0.01	0.01	116.00	0.01	0.13	2.47
100	0.07	0.09	20.88	0.01	0.01	172.01	0.01	0.02	0.44

Table A.1: Average running time for the unconstrained problems.

$c = 3$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01
20	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
30	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
40	0.02	0.02	0.02	0.01	0.01	0.42	0.01	0.01	0.19
50	0.03	0.02	0.02	0.01	0.01	12.33	0.01	0.01	0.20
60	0.03	0.04	0.05	0.01	0.01	65.28	0.01	0.01	0.55
70	0.04	0.05	0.08	0.01	0.01	118.64	0.01	0.01	0.47
80	0.06	0.06	0.10	0.01	0.01	398.29	0.01	0.01	0.64
90	0.06	0.08	0.12	0.01	0.01	228.42	0.01	0.01	0.90
100	0.08	0.07	0.17	0.01	0.01	188.14	0.01	0.01	0.25

$c = 10$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.01	0.01	0.03	0.01	0.01	0.01	0.01	0.01	0.01
20	0.01	0.01	0.03	0.01	0.01	0.01	0.01	0.01	0.17
30	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.19
40	0.03	0.04	0.09	0.01	0.01	0.01	0.01	0.01	0.17
50	0.03	0.02	0.10	0.01	0.01	74.35	0.01	0.01	0.44
60	0.04	0.03	0.16	0.01	0.01	116.69	0.01	0.01	1.24
70	0.05	0.05	0.37	0.01	0.01	380.12	0.01	0.01	2.68
80	0.07	0.06	0.49	0.01	0.01	561.25	0.01	0.01	1.89
90	0.08	0.08	0.62	0.01	0.01	288.02	0.01	0.01	3.21
100	0.08	0.10	0.87	0.01	0.01	545.64	0.01	0.01	0.77

$c = 100$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.11
20	0.01	0.01	0.13	0.01	0.01	0.01	0.01	0.01	0.35
30	0.01	0.01	0.14	0.01	0.01	0.04	0.01	0.01	0.23
40	0.02	0.02	0.13	0.01	0.01	0.22	0.01	0.01	0.99
50	0.02	0.03	0.37	0.01	0.01	6.26	0.01	0.01	4.90
60	0.03	0.04	0.84	0.01	0.01	172.89	0.01	0.01	150.22
70	0.05	0.06	3.44	0.01	0.01	360.01	0.01	0.01	77.16
80	0.05	0.09	3.64	0.01	0.01	432.01	0.01	0.01	73.46
90	0.06	0.09	5.39	0.01	0.01	504.08	0.01	0.01	150.38
100	0.08	0.10	9.83	0.01	0.01	216.02	0.01	0.01	85.12

Table A.2: Average running time for the box-constrained problems.

$\alpha = 0.5n$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.04	0.04	0.09	0.01	0.01	0.01	0.01	0.01	0.01
20	0.16	0.16	0.65	0.01	0.01	0.01	0.01	0.01	0.01
30	0.33	0.44	2.27	0.01	0.01	0.01	0.01	0.01	0.10
40	0.84	1.11	10.02	0.01	0.01	0.01	0.01	0.01	0.01
50	1.79	2.17	34.40	0.01	0.01	0.06	0.01	0.01	0.38
60	2.53	3.88	125.85	0.01	0.01	0.01	0.01	0.01	0.55
70	3.91	5.73	212.39	0.01	0.01	6.53	0.01	0.01	0.07
80	5.55	8.70	888.61	0.01	0.01	72.00	0.01	0.01	0.44
90	7.09	12.10	1130.79	0.01	0.01	0.01	0.01	0.01	0.05
100	10.04	15.72	2929.4	0.01	0.01	216.13	0.01	0.01	0.68

$\alpha = n$

n	MIP			DEBS			HYBRID		
	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$	$\sigma = 0.01$	$\sigma = 0.05$	$\sigma = 0.5$
10	0.40	0.26	0.24	0.01	0.01	0.01	0.01	0.01	0.01
20	22.62	21.23	20.97	0.01	0.01	0.01	0.01	0.01	0.01
30	1744.49	1667.28	1275.03	0.01	0.01	0.01	0.01	0.01	0.01
40	3600	3600	3600	0.01	0.01	0.01	0.01	0.01	0.02
50	3600	3600	3600	0.01	0.01	0.09	0.01	0.01	0.35
60	3600	3600	3600	0.01	0.01	0.37	0.01	0.01	0.18
70	3600	3600	3600	0.01	0.01	1.89	0.01	0.01	0.26
80	3600	3600	3600	0.01	0.01	97.85	0.01	0.01	0.44
90	3600	3600	3600	0.01	0.01	128.69	0.01	0.01	0.43
100	3600	3600	3600	0.01	0.01	203.46	0.01	0.01	0.17

Table A.3: Average running time for the ellipsoid-constrained problems.

A.2 Noise Analysis for the BQPs

The σ values of each problem type

	min	mean	max
Carter	5.71	50.32	96.87
William	5.36	36.94	87.28
bqp50	29.67	55.11	82.06
bqp100	25.60	57.21	97.91
gka	26.38	60.46	128.68
gkb	7.25	17.17	35.41
gkd	44.97	67.75	103.29

Table A.4: The minimum, average, and maximum noise of each problem type of the BQPs.

Bibliography

[1] K. Abhishek, S. Leyffer, and J. Linderoth. Filmint: An outer approximation-based solver for convex mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 22(4):555–567, 2010.

[2] T. Achterberg. Conflict analysis in mixed integer programming. *Discrete Optimization*, 4(1):4–20, 2007.

[3] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007.

[4] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, pages 1–41, 2009.

[5] T. Achterberg and T. Berthold. Hybrid branching. In *Proceedings of Sixth International Conference of the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 309–311, 2009.

[6] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *Information Theory, IEEE Transactions on*, 48(8):2201–2214, 2002.

[7] M. Ajtai, R. Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on*, pages 41–45. IEEE, 2002.

[8] S. B. Akers. Binary decision diagrams. *Computers, IEEE Transactions on*, 27(6):509–516, 1978.

[9] H. R. Andersen, T. Hadzic, J. N. Hooker, and P. Tiedemann. A constraint store based on multivalued decision diagrams. In *International Conference on Principles and Practice of Constraint Programming*, pages 118–132. Springer, 2007.

[10] K. Andersen and A. N. Jensen. Intersection cuts for mixed integer conic quadratic sets. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 37–48. Springer, 2013.

[11] M. F. Anjos, X.-W. Chang, and W.-Y. Ku. Lattice preconditioning for the real relaxation branch-and-bound approach for integer least squares problems. *Journal of Global Optimization*, 59(2-3):227–242, 2014.

[12] A. Atamtürk and M. W. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124, 2005.

[13] E. Bach and J. O. Shallit. *Algorithmic Number Theory: Efficient Algorithms*, volume 1. MIT press, 1996.

[14] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*. Springer, 2001.

[15] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*, volume 39. Springer Science & Business Media, 2012.

[16] R. Barták, M. Salido, and F. Rossi. New trends in constraint satisfaction, planning, and scheduling: a survey. *The Knowledge Engineering Review*, pages 249–279, 2010.

[17] J. C. Beck. Modeling, global constraints, and decomposition. In *Tenth Symposium of Abstraction, Reformulation, and Approximation*, 2013.

[18] J. C. Beck, P. Prosser, and R. J. Wallace. Trying again to fail first. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints*, volume 3419 of *Lecture Notes in Artificial Intelligence*, pages 41–55. Springer-Verlag, 2005.

[19] D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. Hooker. *Decision Diagrams for Optimization*. Springer, 2016.

[20] D. Bergman and J. N. Hooker. Graph coloring facets from all-different systems. In *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems*, pages 50–65. Springer, 2012.

[21] T. Berthold. Rens. *Mathematical Programming Computation*, 6(1):33–54, 2014.

[22] T. Berthold, S. Heinz, and M. E. Pfetsch. Nonlinear pseudo-boolean optimization: relaxation or propagation? In *International Conference on Theory and Applications of Satisfiability Testing*, pages 441–446. Springer, 2009.

[23] T. Berthold, S. Heinz, and S. Vigerske. Extending a cip framework to solve MIQCPs. In *Mixed Integer Nonlinear Programming*, pages 427–444. Springer, 2012.

[24] C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In *In Proceedings of the 19th National Conference on Artificial Intelligence (AAAI04)*, pages 112–117, 2004.

[25] A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1):55–68, 2007.

[26] A. Billionnet, S. Elloumi, and A. Lambert. An efficient compact quadratic convex reformulation for general integer quadratic programs. *Computational Optimization and Applications*, 54(1):141–162, 2013.

[27] R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2012.

[28] A. Bley, A. M. Gleixner, T. Koch, and S. Vigerske. Comparing MIQCP solvers to a specialised algorithm for mine production scheduling. In *Modeling, Simulation and Optimization of Complex Processes*, pages 25–39. Springer, 2012.

[29] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, et al. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

[30] P. Bonami, M. Kilinç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In *Mixed integer nonlinear programming*, pages 1–39. Springer, 2012.

[31] P. Bonami and A. Tramontani. Advances in CPLEX for mixed integer nonlinear optimization. Presented at ISMP 2015, Pittsburgh, PA, 2015.

[32] B. Borchers and J. E. Mitchell. A computational comparison of branch and bound and outer approximation algorithms for 0–1 mixed integer nonlinear programs. *Computers & Operations Research*, 24(8):699–701, 1997.

[33] M. A. Borno. Reduction in solving some integer least squares problems. *arXiv preprint arXiv:1101.0382*, 2011.

[34] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[35] S. Burer and A. N. Letchford. Non-convex mixed-integer nonlinear programming: a survey. *Surveys in Operations Research and Management Science*, 17(2):97–106, 2012.

[36] S. Burer and A. Saxena. The MILP road to MIQCP. In *Mixed Integer Nonlinear Programming*, pages 373–405. Springer, 2012.

[37] M. R. Bussieck and S. Vigerske. MINLP solver software. *Wiley Encyclopedia of Operations Research and Management Science, Wiley, Chichester*, 2010.

[38] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123(2):333–345, 2000.

[39] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.

[40] M. W. Carter. The indefinite zero-one quadratic problem. *Discrete Applied Mathematics*, 7(1):23–44, 1984.

[41] C. Castro and S. Manzano. Variable and value ordering when solving balanced academic curriculum problems. *Proceedings of the ERCIM Working Group on Constraints*, 2001.

[42] X.-W. Chang and G. H. Golub. Solving ellipsoid-constrained integer least squares problems. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1071–1089, 2009.

[43] X.-W. Chang and Q. Han. Solving box-constrained integer least squares problems. *Wireless Communications, IEEE Transactions on*, 7(1):277–287, 2008.

[44] X.-W. Chang and T. Zhou. Miles: Matlab package for solving mixed integer least squares problems. *GPS Solutions*, 11(4):289–294, 2007.

[45] M. Chiarandini, L. Di Gaspero, S. Gualandi, and A. Schaerf. The balanced academic curriculum problem revisited. *Journal of Heuristics*, 18(1):119–148, 2012.

[46] P. R. Cohen. *Empirical methods for artificial intelligence*, volume 139. MIT press Cambridge, 1995.

[47] CPLEX. IBM ILOG CPLEX optimization studio. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html, accessed October 25, 2016.

[48] M. O. Damen, H. El Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *Information Theory, IEEE Transactions on*, 49(10):2389–2402, 2003.

[49] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102(1):71–90, 2005.

[50] R. Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[51] R. Dechter and D. Frost. Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence*, 136(2):147–188, 2002.

[52] F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15(3):404–429, 2010.

[53] N. Downing, T. Feydy, and P. J. Stuckey. Explaining alldifferent. In *Proceedings of the Thirty-fifth Australasian Computer Science Conference-Volume 122*, pages 115–124. Australian Computer Society, Inc., 2012.

[54] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36(3):307–339, 1986.

[55] H. A. Eiselt and C.-L. Sandblom. *Integer programming and network models*. Springer Science & Business Media, 2013.

[56] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985.

[57] G. Finke, R. Burkard, and F. Rendl. Quadratic assignment problems. *Surveys in combinatorial optimization*, page 61, 2011.

[58] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical programming*, 66(1-3):327–349, 1994.

[59] R. Fourer and D. M. Gay. Extending an algebraic modeling language to support constraint programming. *INFORMS Journal on Computing*, 14(4):322–344, 2002.

[60] E. C. Freuder. In pursuit of the holy grail. *Constraints*, 2(1):57–61, 1997.

[61] I. P. Gent, K. E. Petrie, and J.-F. Puget. Symmetry in constraint programming. *Foundations of Artificial Intelligence*, 2:329–376, 2006.

[62] A. M. Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[63] A. M. Gleixner. *Exact and fast algorithms for mixed-integer nonlinear programming.* PhD thesis, Technische Universität Berlin, 2015.

[64] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge. Three enhancements for optimization-based bound tightening. *Journal of Global Optimization*, 2016. online first.

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

[66] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[67] C. Gomes and T. Walsh. Randomness and structure. *Foundations of Artificial Intelligence*, 2:639–664, 2006.

[68] C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of automated reasoning*, 24(1-2):67–100, 2000.

[69] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering*, 3(3):227–252, 2002.

[70] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985.

[71] S. Hammarling and C. Lucas. Updating the QR factorization and the least squares problem. MIMS EPrint: 2008.111, 2008.

[72] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–314, 1980.

[73] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with applications to GPS. *Signal Processing, IEEE Transactions on*, 46(11):2938–2952, 1998.

[74] S. Heinz, W.-Y. Ku, and J. C. Beck. Recent improvements using constraint integer programming for resource allocation and scheduling. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 12–27. Springer, 2013.

[75] S. Heinz and J. Schulz. Explanations for the cumulative constraint: An experimental study. In *International Symposium on Experimental Algorithms*, pages 400–409. Springer, 2011.

[76] S. Heinz, J. Schulz, and J. C. Beck. Using dual presolving reductions to reformulate cumulative constraints. *Constraints*, 18(2):166–201, 2013.

[77] B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Hybrid modelling for robust solving. *Annals of Operations Research*, 130(1-4):19–39, 2004.

[78] B. Hnich, Z. Kiziltan, and T. Walsh. Modelling a balanced academic curriculum problem. In *Fifth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR2002)*, pages 121–131, 2002.

[79] J. Hooker. *Integrated Methods for Optimization, 2nd edition.* Springer, 2012.

[80] S. Jazaeri, A. Amiri-Simkooei, and M. Sharifi. Modified weighted integer least squares estimations for gnss integer ambiguity resolution. *Survey Review*, 46(335):112–121, 2014.

[81] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art.* Springer Science & Business Media, 2009.

[82] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 193–206. ACM, 1983.

[83] G. Katsirelos and F. Bacchus. Generalized nogoods in CSPs. In *In The Twentieth National Conference on Artificial Intelligence (AAAI05)*, pages 390–396, 2005.

[84] M. Khan. Updating inverse of a matrix when a column is added/removed. Technical report, University of British Columbia, Vancouver, 2008.

[85] M. Kisialiou and Z.-Q. Luo. Performance analysis of quasi-maximum-likelihood detector based on semi-definite programming. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 3, pages iii–433. IEEE, 2005.

[86] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. Bixby, E. Danna, G. Gamrath, A. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, pages 1–61, 2011.

[87] N. Krislock, J. Malick, and F. Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Mathematical Programming*, pages 1–26, 2012.

[88] N. Krislock, J. Malick, and F. Roupin. BiqCrunch online solver (2012). http://lipn.univ-paris13.fr/BiqCrunch/solver, Retrieved: 12/22/2013.

[89] W.-Y. Ku and J. C. Beck. Combining discrete ellipsoid-based search and branch-and-cut for binary quadratic programming problems. In *Integration of AI and OR Techniques in Constraint Programming*, pages 334–350. Springer, 2014.

[90] W.-Y. Ku and J. C. Beck. Combining constraint propagation and discrete ellipsoid-based search to solve the exact quadratic knapsack problem. In *Integration of AI and OR Techniques in Constraint Programming*, pages 231–239. Springer, 2015.

[91] W.-Y. Ku and J. C. Beck. Constraint programming for strictly convex integer quadratically-constrained problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 316–332. Springer, 2016.

[92] W.-Y. Ku and J. C. Beck. A mixed integer programming based hybrid algorithm for integer least squares problems for communications applications. *Submitted to European Journal of Operational Research*, 2017.

[93] W.-Y. Ku, T. Pinheiro, and J. C. Beck. CIP and MIQP models for the load balancing nurse-to-patient assignment problem. In *International Conference on Principles and Practice of Constraint Programming*, pages 424–439. Springer, 2014.

[94] J.-M. Kuusinen, J. Sorsa, and M.-L. Siikonen. The elevator trip origin-destination matrix estimation problem. *Transportation Science*, 49(3):559–576, 2014.

[95] T. Lambert, C. Castro, E. Monfroy, and F. Saubion. Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation. In *International Conference on Artificial Intelligence and Soft Computing*, pages 410–419. Springer, 2006.

[96] Y. Lebbah, C. Michel, and M. Rueher. A rigorous global filtering algorithm for quadratic constraints. *Constraints*, 10(1):47–65, 2005.

[97] J. Lee and S. Leyffer. *Mixed integer nonlinear programming*, volume 154. Springer Science & Business Media, 2011.

[98] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[99] L. Létocart, M.-C. Plateau, and G. Plateau. An efficient hybrid heuristic method for the 0-1 exact k-item quadratic knapsack problem. *Pesquisa Operacional*, 34(1):49–72, 2014.

[100] M. Lewis, B. Alidaee, and G. Kochenberger. Using xQx to model and solve the uncapacitated task allocation problem. *Operations research letters*, 33(2):176–182, 2005.

[101] S. Leyffer. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18(3):295–309, 2001.

[102] D. Li, X. Sun, and C. Liu. An exact solution method for unconstrained quadratic 0–1 programming: a geometric approach. *Journal of Global Optimization*, 52(4):797–829, 2012.

[103] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1):193–228, 1998.

[104] S. C. Loong, W.-Y. Ku, and J. C. Beck. Q-bounds consistency for the spread constraint with variable mean. *Constraints*, pages 1–7, 2016.

[105] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1):397–446, 2002.

[106] R. Martí, M. Gallego, and A. Duarte. A branch and bound algorithm for the maximum diversity problem. *European Journal of Operational Research*, 200(1):36–44, 2010.

[107] A. Martin. General mixed integer programming: Computational issues for branch-and-cut algorithms. In *Computational combinatorial optimization*, pages 1–25. Springer, 2001.

[108] J. Maurer, J. Jaldén, D. Seethaler, and G. Matz. Achieving a continuous diversity-complexity tradeoff in wireless mimo systems via pre-equalized sphere-decoding. *IEEE Journal of Selected Topics in Signal Processing*, 3(6):986–999, 2009.

[109] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I: Convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[110] D. Micciancio and P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1468–1480. Society for Industrial and Applied Mathematics, 2010.

[111] M. Milano, G. Ottosson, P. Refalo, and E. S. Thorsteinsson. The role of integer programming techniques in constraint programming's global constraints. *INFORMS Journal on Computing*, 14(4):387–402, 2002.

[112] R. Misener and C. A. Floudas. ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations. *Journal of Global Optimization*, 2014. DOI: 10.1007/s10898-014-0166-2.

[113] H. Mittelmann. Mixed integer linear programming benchmark. http://plato.asu.edu/ftp/milpc.html, accessed October 25, 2016.

[114] J.-N. Monette, N. Beldiceanu, P. Flener, and J. Pearson. A parametric propagator for discretely convex pairs of sum constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–544. Springer, 2013.

[115] J.-N. Monette, P. Schaus, S. Zampelli, Y. Deville, P. Dupont, et al. A CP approach to the balanced academic curriculum problem. In *Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, volume 7, 2007.

[116] C. Mullinax and M. Lawley. Assigning patients to nurses in neonatal intensive care. *Journal of the Operational Research Society*, 53(1):25–35, 2002.

[117] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[118] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[119] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and computational number theory*, 42:75–88, 1990.

[120] O. Ohrimenko, P. J. Stuckey, and M. Codish. Propagation via lazy clause generation. *Constraints*, 14(3):357–391, 2009.

[121] G. Pesant. Achieving domain consistency and counting solutions for dispersion constraints. *INFORMS Journal on Computing*, 27(4):690–703, 2015.

[122] G. Pesant. Balancing nursing workload by constraint programming. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 294–302. Springer, 2016.

[123] G. Pesant, C.-G. Quimper, and A. Zanarini. Counting-based search: Branching heuristics for constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 43(1):173–210, 2012.

[124] G. Pesant and J.-C. Régin. Spread: A balancing constraint based on statistics. In *Principles and Practice of Constraint Programming-CP 2005*, pages 460–474. Springer, 2005.

[125] J. Pryor and J. W. Chinneck. Faster integer-feasibility in mixed-integer linear programs by branching to force change. *Computers and Operations Research*, 38:1143–1152, 2011.

[126] Y. Puranik and N. V. Sahinidis. Domain reduction techniques for global NLP and MINLP optimization. *Constraints*, 2017. online first.

[127] I. Quesada and I. E. Grossmann. An LP/NLP based branch and bound algorithm for convex minlp optimization problems. *Computers & Chemical engineering*, 16(10):937–947, 1992.

[128] C.-G. Quimper, P. Van Beek, A. López-Ortiz, A. Golynski, and S. B. Sadjad. An efficient bounds consistency algorithm for the global cardinality constraint. In *Principles and Practice of Constraint Programming–CP 2003*, pages 600–614. Springer, 2003.

[129] P. Refalo. Tight cooperation and its application in piecewise linear optimization. In *Principles and Practice of Constraint Programming CP99*, volume 1713 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 1999.

[130] F. Rendl, G. Rinaldi, and A. Wiegele. A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 295–309. Springer, 2007.

[131] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010.

[132] E. T. Richards, B. Richards, and I. Parc. Nogood learning for constraint satisfaction. In *Proceedings CP-96 Workshop on Constraint Programming Applications*, 1996.

[133] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*, volume 2. Elsevier Science, 2006.

[134] N. V. Sahinidis. *BARON 14.3.1: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2014.

[135] J. Schaller. Single machine scheduling with early and quadratic tardy penalties. *Computers & Industrial Engineering*, 46(3):511–532, 2004.

[136] P. Schaus. *Solving balancing and bin-packing problems with constraint programming.* PhD thesis, UCL, 2009.

[137] P. Schaus, Y. Deville, P. Dupont, and J.-C. Régin. Simplification and extension of the spread constraint. In *Third international workshop on constraint propagation and implementation*, pages 77–91, 2006.

[138] P. Schaus and J.-C. Régin. Bound-consistent spread constraint. *EURO Journal on Computational Optimization*, pages 1–24, 2013.

[139] P. Schaus, P. Van Hentenryck, and J.-C. Régin. Scalable load balancing in nurse to patient assignment problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 248–262. Springer, 2009.

[140] C.-P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

[141] A. Schutt, T. Feydy, P. J. Stuckey, and M. G. Wallace. Explaining the cumulative propagator. *Constraints*, 16(3):250–282, 2011.

[142] SCIP. Solving constraint integer programs. http://scip.zib.de, accessed October 25, 2016.

[143] H. D. Sherali and W. P. Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.

[144] M. Tawarmalani and N. V. Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, volume 65. Springer Science & Business Media, 2002.

[145] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

[146] P. J. Teunissen, A. Kleusberg, and P. Teunissen. *GPS for Geodesy*, volume 2. Springer Berlin, 1998.

[147] S. Turner, D. Romero, P. Zhang, C. Amon, and T. Chan. A new mathematical programming approach to optimize wind farm layouts. *Renewable Energy*, 63:674–680, 2014.

[148] P. van Emde-Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Mathematisch Instituut, Amsterdam, The Netherlands, 1981.

[149] P. Van Hentenryck and L. Michel. *Constraint-based local search*. The MIT press, 2009.

[150] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica: a modeling language for global optimization*. MIT press, 1997.

[151] P. Van Hentenryck and M. Milano, editors. *Hybrid Optimization: Ten Years of CPAIOR*. Springer, 2011.

[152] W.-J. van Hoeve. The alldifferent constraint: A survey. *arXiv preprint cs/0105015*, 2001.

[153] W.-J. van Hoeve and I. Katriel. Global constraints. *Foundations of Artificial Intelligence*, 2:169–208, 2006.

[154] J. P. Vielma, I. Dunning, J. Huchette, and M. Lubin. Extended formulations in mixed integer conic quadratic programming. *arXiv preprint arXiv:1505.07857*, 2015.

[155] S. Vigerske and A. Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. ZIB-Report 16-24, Zuse Institute Berlin, May 2016.

[156] L. J. Watters. Reduction of integer polynomial programming problems to zero-one linear programming problems. *Operations Research*, 15(6):1171–1174, 1967.

[157] T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex minlp problems. *Computers & Chemical Engineering*, 19:131–136, 1995.

[158] A. Wiegele. Biq mac library–a collection of max-cut and quadratic 0–1 programming instances of medium size. *Preprint*, 2007.

[159] A. Williams. *Quadratic 0-1 Programming Using the Roof Dual: With Computational Results*. RUTCOR, Hill Center, Rutgers University, 1985.

[160] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization—Eureka, You Shrink!*, pages 185–207. Springer, 2003.

[161] X. Xie. *Theory and Algorithms for Some Integer Least Squares Problems*. PhD thesis, McGill University, 2014.

[162] X. Xie, X.-W. Chang, and M. A. Borno. Partial LLL reduction. *Proceedings of IEEE GLOBECOM*, 2011.

[163] P. Y. Zhang, D. A. Romero, J. C. Beck, and C. H. Amon. Solving wind farm layout optimization with mixed integer programs and constraint programs. *EURO Journal on Computational Optimization*, 2(3):195–219, 2014.