A Formal Analysis of the Propagation for the Ellipsoid Global Constraint and an Application to the Selective Tree Breeding Problem

by

Stefana Filipova

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

Abstract

A Formal Analysis of the Propagation for the Ellipsoid Global Constraint and an Application to the Selective Tree Breeding Problem

Stefana Filipova
Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto
2018

The ELLIPSOID global constraint is one of the few global constraints used for reasoning about convex quadratic functions. This thesis compares the strengths of the most successful propagation algorithm for the ELLIPSOID global constraint, namely the BOX propagation, against the standard propagation algorithm based on the expression tree of the mathematical formulation as implemented, for example, in the well-known solver IBM ILOG CP Optimizer. We perform a formal analysis of the strengths of both propagation algorithms, showing that there are conditions under which the BOX propagation algorithm is not as good as the standard propagation algorithm. We apply the ELLIPSOID global constraint to the selective tree breeding problem, a known problem from the forest genetics literature. Our computational study shows that the ELLIPSOID global constraint is not a suitable candidate for this particular problem, which is consistent with the findings from our theoretical analysis.

Acknowledgements

First, I would like to thank my supervisor Professor J. Christopher Beck, for giving me the unique opportunity to do my graduate program under his supervision. In spite of some challenges along the way, with his patience and careful guidance I was able to grow as a researcher and complete this research work. I truly thank him for making me realize how valuable is hard work.

I am forever grateful that I was fortunate enough to work with Professor Merve Bodur, my co-advisor. I learned an incredible amount of valuable things from her, from the flow of mathematical proofs to even good editing in LaTeX. It is very hard to put into words how grateful I am for her encouragement, sincere advice and constant support over the last year of my program.

I would also like to thank my thesis committee members, Professor Timothy Chan and Professor Andre Cire, for taking the time to review the thesis and provide constructive feedback. A big *thank you* goes especially to Professor Timothy Chan, for teaching me the solid foundations of operations research and applied optimization in his course *MIE1620* and constantly inspiring me as an outstanding researcher.

Next, I want to thank my colleague and good friend Ranjith Kumar at the Toronto Intelligent Decision Engineering Lab (TIDEL) who started the MASc journey at the same time with me and went through the same challenges of graduate school. Thank you for all of the fruitful and interesting discussions we had on sports, planes, cultures and what not! And thank you for your friendship when research life got difficult at times. I would also like to thank Wen-Yang Ku for taking all the time in the world (even after graduation!) to discuss the ELLIPSOID global constraint with me. I wish all the best to the rest of the TIDEL members: Kyle, Chiara, Eldan, Arik and Chang.

I would like to thank my wonderful best friends (*my magic beans*) from all around the world for putting up with my continuous complaining about graduate school. At the same time, they have been my biggest fans who never stopped believing in me.

And last but most importantly, I want to express my deepest gratitude to my family back in Macedonia, my dear parents and my brother, for their boundless love and support. Without their many sacrifices, I would not be where I am today.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The main focus of this thesis is to analyze the strength of the constraint propagation algorithms for the ELLIPSOID global constraint in order to either design stronger propagations for quadratic constraints or determine new possible applications for the ELLIPSOID constraint.

The ELLIPSOID global constraint was proposed by Ku & Beck [32] as a new global constraint for strictly convex quadratic functions. Three propagation algorithms were introduced for the ELLIPSOID constraint from which we only focus on analyzing its most successful one: the BOX propagation algorithm. We give a detailed overview of the fundamental concepts in Chapter 2.

Integer quadratically constrained problems (IQCPs) arise in diverse areas such as global positioning systems [59], communications [1] and cryptography [47]. The IQCP is $\mathcal{NP}$-hard problem [65], that is there does not exist a known algorithm which is able to solve this problem in a polynomial time.

Considering the practical significance and theoretical challenge of the IQCP, it is important to develop exact methodologies to efficiently solve this problem. Even though as an $\mathcal{NP}$-hard problem it can be solved with an exhaustive search, it usually takes prohibitively long time to check all feasible solutions for large-scale problems. Consequently, we want to solve IQCP by searching the solution space in a more systematic manner.

The ELLIPSOID global constraint has shown success in a number of problems of the IQCP problem class [32], such as binary quadratic programming problems [68], exact quadratic knapsack problems [35] and ellipsoid-constrained integer least square problems [19]. Despite its success in these problems, it has not yet been formally analyzed why the BOX propagation algorithm for the ELLIPSOID global constraint performed best in terms of running time to prove optimality on particularly these problems. We are interested to know on which problem structures the BOX propagation algorithm shows dominance over the propagation algorithm incorporated into IBM ILOG CP Optimizer.

Another motivation to analyze the propagation algorithms for the ELLIPSOID global constraint is that there are only a few known global constraints in the constraint catalog dedicated for inference with respect to quadratic expressions [7] and so designing stronger propagations for quadratic constraints is very significant. While constraint propagation algorithms have shown success in combinatorial optimization [63], they have not been exhaustively analyzed for particularly for nonlinear constraints. We aim to contribute to this with our analysis.

1.1 Thesis Outline

In Chapter 2, we give an overview of some important classes of optimization problems focusing especially on the IQCP class. We survey the literature of solution methodologies for solving ellipsoid constrained problems with a focus on two techniques, namely mixed-integer programming (MIP) and constraint programming (CP). We especially consider constraint programming as an approach originating from the artificial intelligence community. We also describe the fundamental concepts of the ELLIPSOID global constraint and the propagation algorithms for it that we formally analyze throughout the thesis.

In Chapter 3, we formally analyze the strengths of the two propagation algorithms for the ELLIPSOID global constraint, namely the BOX propagation and the standard propagation algorithm behind IBM ILOG CP Optimizer. There has not been any formal analysis of the strength of the BOX propagation algorithm [32]. Therefore we develop a formal propagation analysis and present detailed proofs for different cases in order to develop a profound understanding of when one propagation algorithm is dominant.

In Chapter 4, we use the ELLIPSOID global constraint for solving the selective tree breeding problem [44]: a problem studied in the tree breeding literature in which a fixed-size breeding population needs to be selected from a list of candidates such that the genetic value of the population is maximized. Our empirical study shows that the ELLIPSOID is not the best choice for solving this problem.

In Chapter 5, we provide concluding remarks and discuss future research directions.

1.2 Summary of Contributions

The contributions of this thesis are as follows.

- In **Chapter 2**, we present a concise and clear explanation of the constraint propagation for ellipsoid constraints behind one of the most well-known CP solvers - IBM ILOG CP Optimizer.

- In **Chapter 3**, we introduce a new formal analysis of the strengths of constraint propagation algorithms for the ELLIPSOID global constraint. Throughout the analysis, we identify some future applications of the ELLIPSOID constraint as well as some possibilities of designing stronger propagation algorithms.

- In **Chapter 4**, we investigate a new application of the ELLIPSOID global constraint for the selective tree breeding problem. We introduce a constraint programming formulation of the problem and show that with further improvements this technique could be competitive to existing technology, at least on some small problems.

Chapter 2

Literature Review

In this chapter, we review the literature on solution techniques for solving ellipsoid constrained problems, with a focus on constraint programming. The purpose of the literature review is to explain the concept of the ELLIPSOID global constraint that we use throughout this thesis and to place it within its context in the optimization literature.

In Section 2.1, we present a simple classification of some important classes of optimization problems. In Section 2.2 we describe the integer quadratically constrained problem (IQCP) through motivating examples from the literature. In Section 2.3 we give a summary of the existing solution techniques used to solve IQCPs. We focus especially on mixed-integer nonlinear programming (MINLP) as a technique from the operations research community and constraint programming (CP) as a technique coming from the artificial intelligence community. In Section 2.4 we explain the ELLIPSOID global constraint and the two propagation algorithms for it that are further analyzed in this thesis.

2.1 Classification of Optimization Problems

In this section, we present several important classes of optimization problems that differ from each other in terms of their level of difficulty and model generality. We give formal definitions for each of them and then discuss the class of problems that are explored in this thesis. We start with the definition of a mathematical optimization problem and the related notions.

Definition 2.1.1 (Optimization Problem [17]). A *mathematical optimization problem* is defined as:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & q_i(\boldsymbol{x}) \leq 0, \;\; i \in \{1, \ldots, m\} \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, $q_i(\boldsymbol{x}) \leq 0, i \in \{1, \ldots, m\}$ are the *constraints* and $\boldsymbol{x} = (x_1, \ldots, x_n)$ is the vector of the *decision variables*. The decision variables represent the entities whose values need to be determined to yield the optimal value for the objective function while satisfying the limitations that the constraints define. A vector $\boldsymbol{x}$ which is satisfying all the constraints is called a *feasible solution* and the set of all feasible solutions is called the *feasible region*. An *optimal solution* to the problem (2.1) is a vector $\boldsymbol{x}^*$ that attains the *smallest* objective value, i.e., optimal value, among the vectors that satisfy the constraints.

In Figure 2.1 we present a graphical overview of some important classes of optimization problems and their classification in terms of level of solution difficulty and model generality which is consistent with the one presented by Pruessner [50]. A difficult optimization problem is defined as a problem for which a general-purpose solver cannot solve large instances reliably and in a reasonable time. Given two optimization problems A and B, we say that A is *more difficult* than B if a general-purpose solver can solve B faster (the time taken to solve) than A and it scales better with the increase in the instance size. On the left side of Figure 2.1 we have the problems which are less difficult to solve and have less general model such as linear programs and quadratic programs. On the right side, the problems are more difficult to solve and have a more general model representation such as second-order cone programs and semidefinite programs. The arrows between problem classes point from a less general and less difficult class of problems to a superset of the more general class of problems. Next, we present the fundamental definitions of each of the problem classes based on the book *Convex Optimization (2004)* [17] and discuss how one class is a subset of another.

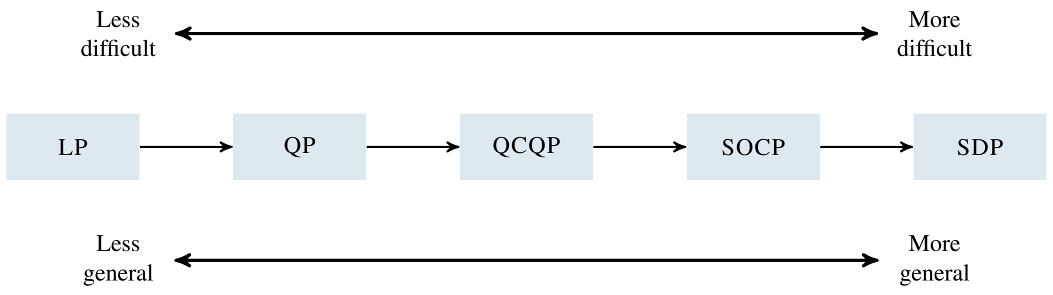


Figure 2.1: A simple classification of optimization problems.

Definition 2.1.2 (Linear Programming [17]). Linear programs are convex optimization problems where the objective and constraints are affine functions. A *linear program* (LP) is defined as:

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} + d \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{c} \in \mathbb{R}^n, \boldsymbol{b} \in \mathbb{R}^m$ and $d \in \mathbb{R}$.

Definition 2.1.3 (Quadratic Programming [17]). Quadratic programs are convex optimization problems where the objective function is quadratic while the constraints are affine. A *quadratic program* (QP) is defined as:

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} + d \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \end{aligned} \tag{2.2}$$

where $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$ and $d \in \mathbb{R}$.

Definition 2.1.4 (Quadratically Constrained Quadratic Programming [17]). Quadratically constrained quadratic programs are convex optimization problems where either the objective function or the constraints can be quadratic. A *quadratically constrained quadratic program* (QCQP) is defined as:

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_0\boldsymbol{x} + \boldsymbol{c}_0^\top \boldsymbol{x} + d_0 \\ \text{subject to} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_i\boldsymbol{x} + \boldsymbol{c}_i^\top \boldsymbol{x} - d_i \le 0, \quad i \in \{1,\dots,k\} \\ & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \end{aligned} \tag{2.3}$$

where $\boldsymbol{H}_i \in \mathbb{R}^{n\times n}, \boldsymbol{c}_i \in \mathbb{R}^n, d_i \in \mathbb{R},\ \forall i \in \{0,1,\dots,k\}, \boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$.

As we see in Figure 2.1, LPs are a subset of the QPs. By having $\boldsymbol{H}_i = 0, \forall i \in \{1,\dots,k\}$ in (2.2), a QP collapses to an LP. Moreover QPs are a subset of QCQPs. If we take $\boldsymbol{H}_i = 0, \forall i \in \{1,\dots,k\}$ in (2.3), then a QCQP collapses to a QP. A class of problems that are more general than QCQPs are second-order cone programs which are defined as follows.

Definition 2.1.5 (Second-Order Cone Programming [17]). Second-order cone programs are convex optimization problems where the objective function is a linear function while the feasible region is the intersection of hyperplanes and the Cartesian product of second-order cones. A *second-order cone program* (SOCP) is defined as:

$$\begin{aligned} \min \quad & \boldsymbol{f}^\top \boldsymbol{x} \\ \text{subject to} \quad & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\|_2 \le \boldsymbol{c}_i^\top \boldsymbol{x} + d_i, \quad i \in \{1,\dots,k\} \\ & \boldsymbol{F}\boldsymbol{x} = \boldsymbol{h} \end{aligned} \tag{2.4}$$

where $\boldsymbol{A}_i \in \mathbb{R}^{m_i\times n}, \boldsymbol{b}_i \in \mathbb{R}^{m_i}, \boldsymbol{c}_i \in \mathbb{R}^n, \boldsymbol{f} \in \mathbb{R}^n, d_i \in \mathbb{R}\ \forall i \in \{1,\dots,k\}$ and $\boldsymbol{F} \in \mathbb{R}^{p\times n}, \boldsymbol{h} \in \mathbb{R}^p$. The constraint $\|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\|_2 \le \boldsymbol{c}_i^\top \boldsymbol{x} + d_i$ is called a *second-order cone constraint* as it geometrically describes a second-order (Lorentz) cone.

Second-order cone programs are more general than QCQPs as well as LPs. Let $\boldsymbol{0}$ be the vector of zeros of the appropriate dimension. If $\boldsymbol{c}_i = \boldsymbol{0},\ i \in \{1,\dots,k\}$ in (2.4) then the SOCP is a QCQP. Moreover, when $\boldsymbol{A}_i = \boldsymbol{0},\ i \in \{1,\dots,k\}$ then the SOCP (2.4) is an LP.

Semidefinite programs are the broadest class of problems, encompassing SOCPs, QCQPs, QPs and LPs, and also one of the most difficult to solve among those convex optimization problems. They are a generalization of

LPs but are richer in expressiveness than the LPs. We define the semidefinite problem as follows.

Definition 2.1.6 (Semidefinite Programming [17])**.** A *semidefinite program* (SDP) has the form:

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{x}_1 \boldsymbol{F}_1 + \cdots + \boldsymbol{x}_n \boldsymbol{F}_n \leq \boldsymbol{G} \end{aligned}$$

where $\boldsymbol{c} \in \mathbb{R}^n$ and $\boldsymbol{G}, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_n \in \mathbb{R}^{m \times m}$ are symmetric positive semidefinite matrices.

In the case that matrices $\boldsymbol{G}, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_n$ in (2.1.6) are all diagonal, then the SDP reduces to a general LP.

In this thesis we focus on the QCQPs where all the variables are restricted to take integer values. This set of problems is called Integer Quadratically Constrained Programs. They are an important class of problems that includes LPs and QPs and they are also a subset of a more general class of problems such as SOCPs and SDPs.

2.2 Integer Quadratically Constrained Problems (IQCP)

In this section, we first present the formal problem definition of IQCP and then illustrate their vast application through two motivating examples.

2.2.1 Problem Definition

The integer quadratically constrained program (IQCP) is an important class of optimization problems where either the objective function or the constraints are modelled with quadratic functions and all of the variables are restricted to take on integer values. In (2.5) we present the formal IQCP formulation. We consider $\boldsymbol{H}_i, \ \forall i \in \{0, 1, \ldots, k\}$ to be symmetric positive definite matrices.

Definition 2.2.1. (Integer Quadratically Constrained Problems [65]) An *integer quadratically constrained problem* has the form:

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_0 \boldsymbol{x} + \boldsymbol{c}_0^\top \boldsymbol{x} + d_0 \\ \text{subject to} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_i \boldsymbol{x} + \boldsymbol{c}_i^\top \boldsymbol{x} \leq d_i, \quad i \in \{1, \ldots, k\} \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{Z}^n \end{aligned} \tag{2.5}$$

where $\boldsymbol{A} \in R^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m, \boldsymbol{H}_i \in \mathbb{R}^{n \times n}, \boldsymbol{c}_i \in \mathbb{R}^n$ and $q_i \in \mathbb{R}, \forall i \in \{0, 1 \ldots, k\}$.

2.2.2 Motivating Examples

Wind Farm Layout Optimization (WFO) [64]. In order to exemplify the application of IQCPs, we present the problem of wind farm layout optimization [64]. The wind farm optimization (WFO) problem aims to find the optimal placement of wind turbines in the wind farm in order to maximize the power generated by the turbines. The problem is defined with a set of possible locations for the turbines, I, and the number of wind turbines that need to be placed, C. The decision variable $\boldsymbol{x}_i$ takes value of 1 if a wind turbine is placed at position $i \in I$ and 0 otherwise.

Then, WFO problem is modelled as an IQCP as follows:

$$\max \quad \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_j \boldsymbol{H}_{ij} \tag{2.6a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \boldsymbol{x}_i = C \tag{2.6b}$$

$$\boldsymbol{x}_i \in \{0,1\}, \forall i \in I \tag{2.6c}$$

The objective of maximizing the power generated is represented with the quadratic function in (2.6a). The matrix $\boldsymbol{H}$ is called an *interaction* matrix, where the terms H_{ij} represent the power produced as a result of an *interaction* between a wind turbine placed at position $i \in I$ and a wind turbine placed at position $j \in I$. Lastly, constraint (2.6b) ensures that exactly C turbines are placed.

Uncapacitated Task Allocation Problem (UTA) [36]. Information systems tend to look for efficient methods for making use of the availability of powerful parallel and distributed systems [36]. An important issue for the efficient operations of these systems is the optimal task allocation to processors. Given a set of tasks and a set of processors, the uncapacitated task allocation problem (UTA) aims to assign tasks to processors at minimal cost.

The problem is defined with a set of N tasks, a set of M processors and a set $\mathcal{C} = \{(t,k)\}$ where t and k are tasks that require communication during processing. Furthermore f_{ij} denotes the execution cost of task i on processor j, while c_{tk} is the communication cost between task t and task k if they are assigned to different processors. The decision variable $\boldsymbol{x}_{ij}$ takes a value of 1 if task i is assigned to processor j and 0 otherwise.

Then, an IQCP model for UTA problem is given as follows:

$$\min \quad \sum_{i=1}^{N}\sum_{j=1}^{M} f_{ij}\boldsymbol{x}_{ij} + \sum_{(t,k)\in\mathcal{C}} \sum_{j=1}^{M} c_{tk}\boldsymbol{x}_{tj}(1-\boldsymbol{x}_{kj}) \tag{2.7a}$$

$$\text{subject to} \quad \sum_{j=1}^{M} \boldsymbol{x}_{ij} = 1, \quad \forall i \in \{1,\dots,N\} \tag{2.7b}$$

$$\boldsymbol{x}_{ij} \in \{0,1\}, \quad \forall i \in \{1,\dots,N\}, \forall j \in \{1,\dots,M\} \tag{2.7c}$$

Constraint (2.7b) ensures that we assign each task to exactly one processor. In the *quadratic* objective function (2.7a) we minimize the cost of assigning tasks to processors as well as the communication costs between tasks that are assigned to different processors.

2.3 Solution Techniques for IQCPs

In this section we survey exact methods for solving IQCPs. Particularly we focus on two mature technologies, one from operations research (OR) and another from artificial intelligence (AI), namely mixed-integer nonlinear programming and constraint programming.

2.3.1 Mixed Integer Nonlinear Programming (MINLP)

Mixed-integer programming (MIP) is a mathematical optimization approach used for solving optimization problems where the decision variables can take either integer or fractional values. MIP originated from the 1950s [46], and has been widely used within the operations research community ever since. Many MIP techniques are embedded in modern-day MIP solvers which are also used to solve convex mixed-integer quadratically constrained problems (MIQCP), an important subset of mixed-integer nonlinear problems (MINLP) [15]. In this section, we first review the basics of MIP and then describe the fundamental concepts of MINLP methodology.

Definition 2.3.1 (Mixed-Integer Program [46])**.** A *mixed-integer program* (MIP) is a mathematical optimization problem of the form:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) && (2.8a)\\ \text{subject to} \quad & h_i(\boldsymbol{x}) \le 0, \forall i \in \{1, \dots, k\} && (2.8b)\\ & \boldsymbol{x}_j \in \mathbb{Z}, \forall j \in J && (2.8c)\\ & \boldsymbol{x} \in \mathbb{R}^n && (2.8d) \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, $h_i : \mathbb{R}^n \to \mathbb{R}$ are the *constraint functions*, and $J \subseteq \{1, \dots, n\}$ is the index set of the *integer* variables.

Furthermore, we define the continuous relaxation of a mixed-integer program which is obtained when the integrality restrictions on the variables in (2.8d) are dropped. The continuous relaxation is critical to solving MIP as it represents an approximation of the problem that is easier to solve.

Definition 2.3.2 (Continuous Relaxation [46].)**.** The continuous relaxation of a mixed-integer program is defined as:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) && (2.9a)\\ \text{subject to} \quad & h_i(\boldsymbol{x}) \le 0, \forall i \in \{1, \dots, k\} && (2.9b)\\ & \boldsymbol{x} \in \mathbb{R}^n && (2.9c) \end{aligned}$$

We also define *mixed-integer linear program* and *mixed-integer quadratically constrained program*. A mixed-integer program is a *mixed-integer linear program* when the objective function and the constraints are linear functions.

Definition 2.3.3 (Mixed-Integer Linear Program [46])**.** A *mixed-integer linear program* (MILP) is a mathematical optimization problem of the form:

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \boldsymbol{x} && (2.10a)\\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} && (2.10b)\\ & \boldsymbol{x}_j \in \mathbb{Z}, \forall j \in J && (2.10c)\\ & \boldsymbol{x} \in \mathbb{R}^n && (2.10d) \end{aligned}$$

where $\boldsymbol{c} \in \mathbb{R}^n, \boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m$ and $J \subseteq \{1, \dots, n\}$ is the index set of the *integer* variables.

Note that the continuous relaxation of an MILP is an LP.

A mixed-integer program is a *mixed-integer quadratically constrained program* when the objective function or one of the constraint functions is a quadratic function.

Definition 2.3.4 (Mixed-Integer Quadratically Constrained Program [34])**.** A *mixed-integer quadratically constrained program* (MIQCP) is a mathematical optimization problem of the form:

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_0 \boldsymbol{x} + \boldsymbol{c}_0^\top \boldsymbol{x} + d_0 \\ \text{subject to} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_i \boldsymbol{x} + \boldsymbol{c}_i^\top \boldsymbol{x} - d_i \le 0, \forall i \in \{0, 1, \dots, k\} \\ & \boldsymbol{x}_j \in \mathbb{Z}, \forall j \in J \\ & \boldsymbol{x} \in \mathbb{R}^n \end{aligned}$$

where $\boldsymbol{H}_i \in \mathbb{R}^{n\times n}, \boldsymbol{c}_i \in \mathbb{R}^n, d_i \in \mathbb{R}, \quad \forall i \in \{0, 1, \dots, k\}$ and $J \subseteq \{1, \dots, n\}$ is the index set of the *integer* variables.

Mixed-integer nonlinear programs are a broader class of optimization problems than the MILP and MIQCP where some of the variables can take integer values while the constraint functions and objective function are modeled with nonlinear functions [34]. Such problems arise in diverse engineering problems, where discrete decisions are made while the systems are described by nonlinear dynamics. Some of their important engineering applications involve transmission switching [22], optimal control [55], network design with queuing delay constraints [16], optimizing flow in gas networks [39] and resource allocation for homeland security [13] .

Definition 2.3.5 (Mixed-Integer Nonlinear Program [14])**.** A *mixed-integer nonlinear program* (MINLP) is defined as:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & h_i(\boldsymbol{x}) \leq 0, \forall i \in \{1,\dots,k\} \\ & \boldsymbol{x}_j \in \mathbb{Z}, \forall j \in J \\ & \boldsymbol{x} \in \mathbb{R}^n \end{aligned} \tag{MINLP}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function* that can be a nonlinear function, $h_i : \mathbb{R}^n \to \mathbb{R},\ i \in \{1,\dots,k\}$ are the *constraint functions* which can be *nonlinear* and $J \subseteq \{1,\dots,n\}$ is the index set of the *integer* variables.

MINLPs can be solved by performing a *linearization* of the MINLP objective function since the objective can be a nonlinear function. This linearization is done by introducing an auxiliary variable, denoted by α, representing the objective function value and moving the original objective function to the constraints.

Definition 2.3.6 (MINLP Objective Linearization [14])**.** A MINLP objective linearization (MINLP$_L$) is defined as follows:

$$\begin{aligned} \min \quad & \alpha \\ \text{subject to} \quad & f(\boldsymbol{x}) \leq \alpha \\ & h_i(\boldsymbol{x}) \leq 0, \forall i \in \{1,\dots,k\} \\ & \boldsymbol{x}_j \in \mathbb{Z}, \forall j \in J \\ & \boldsymbol{x} \in \mathbb{R}^n \end{aligned}$$

Another important concept to solving MINLPs is the continuous relaxation of a MINLP. Similar to the notion introduced for MIP, the continuous relaxation of an MINLP is obtained by dropping the integrality restriction on the variables.

Definition 2.3.7 (Continuous relaxation of MINLP [14])**.** The continuous relaxation of an MINLP (MINLP$_R$) is defined as:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & h_i(\boldsymbol{x}) \leq 0, \forall i \in \{1,\dots,k\} \\ & \boldsymbol{x} \in \mathbb{R}^n \end{aligned} \tag{MINLP$_R$}$$

In the past 30 years there has been a growing development and implementation of algorithms for solving MINLPs [15]. Successful solution frameworks and enhancements such as branch-and-bound, outer approximation and cutting planes have been efficiently implemented in a number of state-of-the-art MINLP solvers. Next, we review the branch-and-bound algorithm used within MINLP methodology.

The MINLP Branch-and-Bound method (MINLP-B&B). Branch-and-bound (B&B) is a well-known solution technique first developed by Land and Doig [33]. The branch-and-bound framework is a divide-and-conquer approach.

The step of *branching* (dividing) is defined as partitioning the current search space of the problem into smaller subsets. In other words, the problem is broken down to a set of smaller problems, each being called a subproblem. More specifically, when a fractional optimal solution is found $\tilde{x} \notin \mathbb{Z}^n$, then an index $j \in \{1, \ldots, n\}$ with $\tilde{x}_j \notin \mathbb{Z}$ is chosen and by adding the constraints $x_j \geq \lceil \tilde{x}_j \rceil$ and $x_j \leq \lfloor \tilde{x}_j \rfloor$ two subproblems are created. The step of *bounding* is done by solving the continuous relaxation of a subproblem and obtaining a lower bound. We then compare the lower bound with the upper bound on the optimal value. If the lower bound exceeds or equals to the upper bound, the subproblem cannot lead to a solution with a better cost than the current best solution thus can be eliminated from consideration. The algorithm creates a branch-and-bound search tree, where the term *node* corresponds to a subproblem. The step of eliminating a subproblem from consideration is called *pruning* a node in the search tree.

Next we describe the basic idea of the B&B algorithm. At the beginning, the algorithm finds a solution to the continuous relaxation (MINLP_R) of the original problem (MINLP). The solution from the relaxation corresponds to a lower bound on the optimal value of the original problem. If the solution from the relaxation is found to be feasible to the original problem, then it is also optimal. Otherwise, the algorithm continues with partitioning the problem into subproblems by imposing additional constraints, i.e., performs branching. The algorithm keeps a list ($\mathcal{L}$) of all subproblems and stops when there are no more subproblems to explore. The other stopping criteria is reaching an optimality gap within a user-defined threshold value. The pseudocode for the B&B method for solving MINLP is given in Algorithm 1, where $\boldsymbol{x}^*$ is the current solution, and ub denotes the upper bound on the optimal value. In line 1, the list of nodes is initialized with the original problem, while the upper bound on the optimal solution value is initialized to infinity and the current best solution does not have an initial value. In line 2, we check to see if the list of nodes is empty; if there are no more subproblems in the list to explore, the current solution is optimal. Next in line 3, we select a problem from the list and solve its continuous relaxation in order to obtain the lower bound. Four possibilities can occur based on the solution to the relaxation. In option a.), if the problem (MINLP_R) is infeasible, we prune the node and go to line 2. In option b.), if we obtain a better objective value than the upper bound and the solution is also feasible to the original problem, then we have a new current best solution and the new current best upper bound. In option c.), if the relaxation solution is not better than the current solution then the node is pruned and we go to line 2 again. And lastly, for option d.), in case we obtain a fractional solution we branch and create two new subproblems which are added to the list and go to line 2.

Algorithm 1: The **MINLP-B&B** method [34].

```
1 Initialize $\mathcal{L} \leftarrow$ (MINLP), $ub = \infty, \boldsymbol{x}^* \leftarrow NONE$
2 Check to see if $\mathcal{L} = \emptyset$. If so, then the solution $\boldsymbol{x}^*$ is optimal.
3 Select a problem from the list $\mathcal{L}$ and solve its (MINLP$_R$) to obtain its lower bound $l_R$ and solution $x_R$.
    a. If (MINLP$_R$) is infeasible, prune the node. Go to Step 2.
    b. If $l_R < ub$ and the solution is feasible to the original (MINLP), then set $ub = l_R$, update $\boldsymbol{x}^*$ to $x_R$ and
       prune the node (by bound). Go to Step 2.
    c. If $l_R \geq ub$, then prune the node (by bound). Go to Step 2.
    d. If the solution $x_R$ is fractional then continue with branching by creating two new nodes (subproblems).
       Add all the new subproblems to $\mathcal{L}$. Go to Step 2.
```

2.3.2 Constraint Programming (CP)

Constraint programming (CP) is a discrete optimization technique used to solve combinatorial search problems [10]. It has been an attractive research area since the 1970s as a result of the expressive language it provides for formulating a problem. In addition to general mathematical expressions used for problem formulation, with constraint programming it is possible to describe the problem with symbolic constraints which encapsulate more details about the given problem. In fact, the main concept of CP is to represent a real-world problem accurately in terms of constraints and variables and then to find an assignment of values to the decision variables that satisfy the constraints. For instance, in scheduling, the decision variables can be start times or durations of jobs and constraints can be the availability of machines (or resources) for a limited time.

Similar to MIP, CP is also a search-tree framework where the original problem is represented with the root node and every interior node corresponds to a subproblem derived from assigning values to a subset of the variables. In general, CP is trying to solve $\mathcal{NP}$-complete problems by searching the solution space in a systematic manner. In order to have an efficient performance and not check every possible assignment, the exponential search space needs to be reduced. The way this reduction of the search space is done in CP is through inference. At each node in the tree, *constraint propagation* (an inference technique) is invoked to infer new bounds and/or constraints on the variables, and these changes are updated throughout the other constraints in order to reduce the domains of the variables and cut down the search space further. Even though this technique can find an optimal solution with systematic search, still for large problems it may take a rather prohibitive time [21].

Due to the core strengths of CP such as rich modelling language and strong constraint propagation, it has been applied to a wide range of domains such as scheduling [6, 63], planning [62] and vehicle routing [21, 38].

Next, we focus on the details of the main mechanism used to reduce the search space in CP, which is *constraint propagation*.

Propagation Algorithms. Constraint propagation is a fundamental component of any constraint programming solver. It appears under many different names in the literature such as filtering algorithms [52], constraint inference [48] and local consistency enforcing [29]. The propagation algorithm filters variable domains and removes inconsistent values that do not participate in a solution. For example, consider a simple problem that is enforcing the constraint $x_1 > x_2$, where the domains of the variables are defined as $x_1 \in \{2, 4, 6\}$ and $x_2 \in \{3, 5, 7\}$. By *propagating* this constraint, we can infer that the value 2 from the domain of x_1 and the value 7 from the domain of x_2 do not participate in any solution. Therefore after the propagation step the domains of the variables are changed to $x_1 \in \{4, 6\}$ and $x_2 \in \{3, 5\}$, meaning that the solution space to be explored has been reduced. This example illustrates that values can be removed from variable domains by showing that they do not satisfy a constraint, meaning that they are not feasible to the problem as all the constraints must be satisfied. Hence, constraint propagation is local to a constraint. In general, filtering algorithms are based on an established *local consistency* and an associated polynomial enforcing algorithm developed to transform a constrained problem into a network which satisfies all the constraints [10].

In order to characterize different propagation algorithms, we next describe the classical local consistencies and present fundamental notions for CP, based on the book *Handbook of Constraint Programming (2006)* [10].

Definition 2.3.8 (Constraint network [10])**.** A **constraint network** is a triple $< X, \mathcal{C}, \mathcal{D} >$ described with a finite set of variables $X = \{x_1, \dots, x_n\}$, a set of constraints $\mathcal{C} = \{C_1, \dots, C_m\}$ enforced on the variables and a set of variable domains $\mathcal{D} = \{D(x_1) \times \cdots \times D(x_n)\}$ where $D(x_i)$ is the set of values that can be assigned to x_i.

Definition 2.3.9 (Constraint [10])**.** A **constraint** $C_m \in \mathcal{C}$ is a relation that maps *tuples* of values of the variables to the set $\{$*true*,*false*$\}$.

Definition 2.3.10 ([10])**.** Given a constraint network $< X, \mathcal{C}, \mathcal{D} >$:

- An **assignment** is a function which maps variables to values, i.e. $a : \{x_1, \dots, x_n\} \to \mathbb{R}$.
- An assignment a is **valid** if $a(x_i) \in D(x_i), \ \ \forall i \in \{1, \dots, n\}$.
- A **solution to a constraint** $C_m \in \mathcal{C}$ is an assignment that C_m returns *true*.

The basic propagation mechanism used in most constraint programming solvers is one of the many algorithms for enforcing arc consistency [10]. (Generalized) Arc Consistency is a simple concept that guarantees every value in the domain to be consistent with every constraint individually, and it is defined as follows.

Definition 2.3.11 (Generalized Arc Consistency (GAC) [10])**.** Given a constraint network $< X, \mathcal{C}, \mathcal{D} >$:

- A constraint $C_m \in \mathcal{C}$ is **(generalized) arc consistent** with respect to the variable domains if for every variable $x_i \in X$ and each value $v_i \in D(x_i)$, there exist values $v_j \in D(x_j),\ j \in \{1, \dots, n\} \backslash \{i\}$ such that the constraint $C_m(v_1, \dots, v_n)$ holds true.
- We say that the constraint network is GAC *iff* every constraint is GAC with respect to the variable domains.

Let $\mathcal{A}_m$ be the set of assignments that $C_m \in \mathcal{C}$ maps to *true*, $p(D)$ be the obtained variable domains after running the propagation algorithm p on the variable domain D and $\mathcal{D}$ is the set of all possible variable domains. Given this notation, we formally define *constraint propagation* as follows.

Definition 2.3.12 (Constraint Propagation [56])**. Constraint propagation** is a function which maps variable domains to variable domains, i.e., $p : \mathcal{D} \to \mathcal{D}$ such that $p(D) \subseteq D$ if $D \subseteq \mathcal{D}$ and $p(\hat{D}) \subseteq \tilde{D}$ if $\hat{D} \subseteq \tilde{D} \subseteq \mathcal{D}$.

Finally we define an important property of constraint propagation which is relevant for our analysis in Chapter 3: the soundness of a constraint propagation. Our definition of sound constraint propagation is close to the one defined in [58]. One direct and important consequence from soundness is that a constraint propagation can never remove values for the variables that participate in a solution.

Definition 2.3.13 (Sound Propagation [58])**.** A constraint propagation p is **sound** for a constraint C_m *iff* $p(\mathcal{A}_m) \subset p(\mathcal{D})$.

Global Constraints. The inference algorithms used in CP are often captured in *global constraints* [66]. Global constraints are a powerful CP tool as they provide excellent problem insights and strong filtering of variable domains based on problem structure. In order to explain the concept of global constraints, we describe one of the best-known global constraints, namely ALLDIFFERENT where each of the variables need to take on different values. The ALLDIFFERENT constraint is defined as follows:

Definition 2.3.14. Let $\{x_1, x_2 \ldots, x_n\}$ be a finite set of variables. Then

$$\text{ALLDIFFERENT}(x_1, \ldots, x_n) = \{(v_1, \ldots, v_n) \mid v_i \in D(x_i)\ \forall i \in \{1, \ldots, n\}, v_i \neq v_j\ \forall i, j \in \{1, \ldots, n\}, i \neq j\}$$

In order to show the concept of a global constraint, in Figure 2.2 we present an instance of the graph coloring problem.

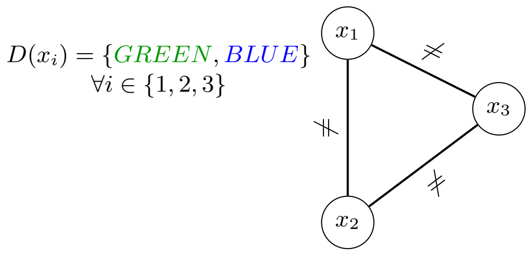


Figure 2.2: A graph coloring example.

In this instance we have three variables (nodes) that are allowed to take one of the two possible values of blue and green color. The edges between the nodes represent the constraints that each pair of adjacent nodes must take on different values, i.e., $x_i \neq x_j$ for all edges (i, j) in the graph. The set of constraints can be *captured* with the ALLDIFFERENT$(\cdot)$ constraint in the following manner:

$$\left.\begin{array}{c} x_i \in \{GREEN, BLUE\},\ i \in \{1,2,3\} \\ \text{CONSTRAINTS}: x_i \neq x_j \ \ \forall \ \ (i,j) \end{array}\right\} \begin{array}{c} x_i \in \{GREEN, BLUE\},\ i \in \{1,2,3\} \\ \text{ALLDIFFERENT}(x_1, x_2, x_3) \end{array}$$

The ALLDIFFERENT$(\cdot)$ is associated with its own filtering algorithm that enforces GAC on the global constraint. In contrast to arc consistency propagation on the group of non-equals constraints which does not derive anything, the GAC on the ALLDIFFERENT$(\cdot)$ representation provides stronger inference by inferring an empty domain for one of the variables, hence showing that there is no solution.

Therefore, we define a global constraint as follows.

Definition 2.3.15 (Global Constraint [53])**.** A **global constraint** is defined as the conjunction of the set of constraints that describe a part of a given problem: $\mathcal{G} = \bigwedge_{i \in \{1,\ldots,m\}} C_i$. It is associated with its own filtering algorithm which often tries to enforce GAC on $\mathcal{G}$.

Global constraints demonstrate many advantages. First, they describe the problem in a more convenient manner with one constraint encapsulating a set of constraints rather than defining each of the independent constraints [53]. Secondly, it is much easier and convenient to guide the search and inference from the simultaneous presence of a set of constraints [11]. Lastly, specific filtering algorithms for global constraints make it possible to incorporate powerful operations research techniques considering a set of constraints as a whole [32].

2.4 The Ellipsoid Global Constraint

The ELLIPSOID global constraint was introduced by Ku & Beck [32] as a novel global constraint that reasons about convex quadratic functions used in ellipsoid constrained problems. First, we describe the constraint for axis-aligned ellipsoids and the BOX propagation algorithm behind it, and then present the same notions for a rotated ellipsoid.

Definition 2.4.1 (ELLIPSOID Constraint for Axis-Aligned Ellipsoid [32])**.** The ELLIPSOID constraint is used for inference in strictly convex quadratic functions. It comprises of a set of variables $\{x_1, \ldots, x_n\}$, a full column rank matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, a vector $\boldsymbol{y} \in \mathbb{R}^n$ and a constant $\beta \in \mathbb{R}$ [32]. For an axis-aligned ellipsoid the **A** matrix is a *diagonal* matrix i.e., $a_{ij} = 0$, $\forall i \neq j$. The constraint ensures the following expression:

$$\sum_{i=1}^{n} (y_i - a_{ii}x_i)^2 \leq \beta \tag{2.11}$$

Note that the constraint (2.11) geometrically describes a hyper-ellipsoid with a center at $\boldsymbol{A}^{-1}\boldsymbol{y}$ as presented in Figure 2.3. The global constraint is denoted as ELLIPSOID$(\{x_1, \ldots, x_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta)$.

Remark 1. The matrix **A** is *positive definite* for strictly convex quadratic constraints i.e., $a_{ii} > 0, \forall i \in \{1, \ldots, n\}$.

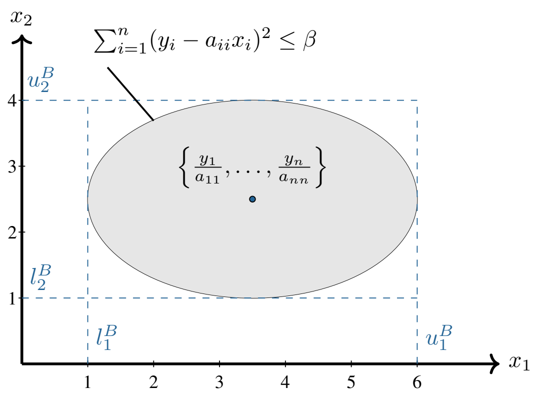

$\sum_{i=1}^{n}(y_i - a_{ii}x_i)^2 \leq \beta$
$\left\{\frac{y_1}{a_{11}}, \dots, \frac{y_n}{a_{nn}}\right\}$
u_2^B
l_2^B
l_1^B
u_1^B
x_1
x_2

Figure 2.3: An example of axis-aligned ellipsoid.

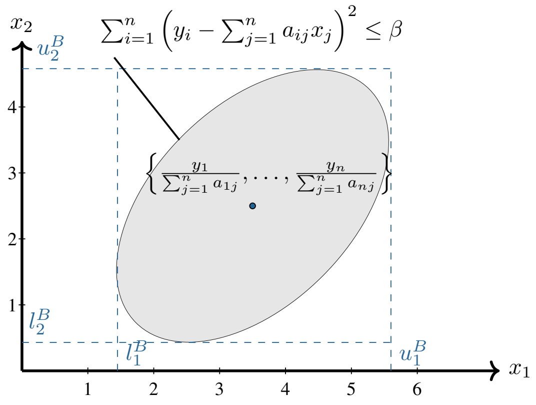

$\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2 \leq \beta$
$\left\{\frac{y_1}{\sum_{j=1}^{n} a_{1j}}, \dots, \frac{y_n}{\sum_{j=1}^{n} a_{nj}}\right\}$
u_2^B
l_2^B
l_1^B
u_1^B
x_1
x_2

Figure 2.4: An example of rotated ellipsoid.

BOX Propagation Algorithm for Axis-Aligned Ellipsoid Constrained Problems. Suppose that variable x_j has variable domains $[l_j, u_j], \forall j \in \{1, \dots, n\}$. The BOX propagation [32] reduces and updates the domains of the variables by computing the *tangent box* of the axis-aligned hyper-ellipsoid (2.11), where the edges of the computed box are parallel to the axes of the coordinate system and intersect with the hyperellipsoid at exactly one point. The lower bound l^B and the upper bound u^B vectors that define the tangent box are computed by solving the following two problems

$$l_j^B := \min_{x \in \mathbb{R}^n} x_j \quad \text{subject to} \quad \sum_{i=1}^{n}(y_i - a_{ii}x_i)^2 \leq \beta$$

$$u_j^B := \max_{x \in \mathbb{R}^n} x_j \quad \text{subject to} \quad \sum_{i=1}^{n}(y_i - a_{ii}x_i)^2 \leq \beta$$

$\forall j \in \{1, \dots, n\}$.

These problems are solved in an efficient way with the algorithm that Chang & Golub [19] present. The compact solutions to the problems are given with the following equations:

$$l_j^B := -\sqrt{\beta}\|A^{-\top}e_j\|_2 + e_j^\top A^{-1}y \tag{2.12}$$

$$u_j^B := \sqrt{\beta}\|A^{-\top}e_j\|_2 + e_j^\top A^{-1}y \tag{2.13}$$

where $e_j \in \mathbb{R}^n$ is the unit vector in the j-th direction, i.e., the j-th column of an $n \times n$ identity matrix.

The BOX propagation computes the new (updated) domain of variable x_j denoted with $[l_j^{\text{BOX}}, u_j^{\text{BOX}}]$ as the intersection of the original bounds $[l_j, u_j]$ and the bounds of the tangent box $[l_j^B, u_j^B]$, i.e.,

$$[l_j^{\text{BOX}}, u_j^{\text{BOX}}] = [l_j, u_j] \bigcap [l_j^B, u_j^B].$$

Next, we give the definition of the ELLIPSOID global constraint and describe how the BOX propagation works for rotated ellipsoids.

Definition 2.4.2 (ELLIPSOID Constraint for Rotated Ellipsoid [32])**.** For a rotated ellipsoid the off-diagonal entries of **A** are not necessarily all zero. The ELLIPSOID constraint then ensures the following expression:

$$\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2 \leq \beta \tag{2.14}$$

The constraint (2.14) geometrically describes a hyper-ellipsoid with a center at $\boldsymbol{A}^{-1}\boldsymbol{y}$ as presented in Figure 2.4. The global constraint is denoted as $\text{ELLIPSOID}(\{x_1, \ldots, x_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta)$

BOX Propagation for Rotated Ellipsoid Constrained Problems. The BOX propagation reduces and updates the domains of the variables by computing the tangent box of the *rotated* ellipsoid. The lower bound l^B and upper bound u^B vectors that define the tangent box can be computed by solving the following optimization problems for each dimension $p \in \{1, \ldots, n\}$, which are constrained with a rotated ellipsoid (2.14) :

$$l_p^B := \min_{x \in \mathbb{R}^n} x_p \quad \text{subject to} \quad \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2 \leq \beta$$

$$u_p^B := \max_{x \in \mathbb{R}^n} x_p \quad \text{subject to} \quad \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2 \leq \beta$$

Then, the updated domain of variable x_p denoted with $[l_p^{\text{BOX}}, u_p^{\text{BOX}}]$ is therefore the intersection of the original domain $[l_p, u_p]$ and the bounds of the tangent box $[l_p^B, u_j^B]$, i.e,

$$[l_p^{\text{BOX}}, u_p^{\text{BOX}}] = [l_p, u_p] \bigcap [l_p^B, u_p^B].$$

2.5 Constraint Propagation Behind IBM ILOG CP Optimizer (CP$_{\text{DEFAULT}}$)

Modern-day CP Solvers such as IBM ILOG CP Optimizer use a specific propagation algorithm for specific types and forms of constraints. In order to illustrate how this specific propagation algorithm (referred to as CP$_{\text{DEFAULT}}$) for the axis-aligned ELLIPSOID constraint (2.11) behind the IBM ILOG CP Optimizer works, we present a small example. For a clear comparison we consider the following ellipsoid constraint that matches the form of the general ellipsoid constraint for axis-aligned ellipsoids (2.11), and its box constraints:

$$(2 - x_1)^2 + (5 - 2x_2)^2 \leq 8, \qquad x_1 \in [0, 5],\ x_2 \in [0, 4] \tag{2.15}$$

First, we represent the quadratic expression for the ellipsoid constraint using an expression tree [26] where the internal nodes represent operators or functions and the leaf nodes denote the variables. The reduction on the variable bounds is done through the expression tree propagation, i.e., CP$_{\text{DEFAULT}}$ propagation algorithm, by using the bounds on the variables to improve the bounds on the expressions and *vice versa*. The constraint propagation algorithm enforces arc consistency throughout the expression tree on a triple of variables, a node with two children, and the constraint connecting them. We discuss the reasoning for this particular constraint through two concepts, namely forward and backward propagation as shown in Figure 2.5 and Figure 2.6, respectively.

Forward Propagation. In Figure 2.5, the forward propagation starts at the leaf nodes of the expression tree and continues until the root node, reasoning the same way about each triple of variables, a node with two children, and the connecting expression. Using the bounds of the children nodes, it finds the *min* and *max* values of the expression as the bounds of the parent node. In our example, the expressions connecting the leaf nodes are x_1 and $2x_2$. The lower and upper bounds for the expression x_1 are updated such that *lower bound* = $\min\{x_1 \mid x_1 \in [0,5]\}$ and *upper bound* = $\max\{x_1 \mid x_1 \in [0,5]\}$. In a similar manner, the bounds on the expression $2x_2$ are updated with *lower bound* = $\min\{2x_2 \mid x_2 \in [0,4]\}$ and *upper bound* = $\max\{2x_2 \mid x_2 \in [0,4]\}$.

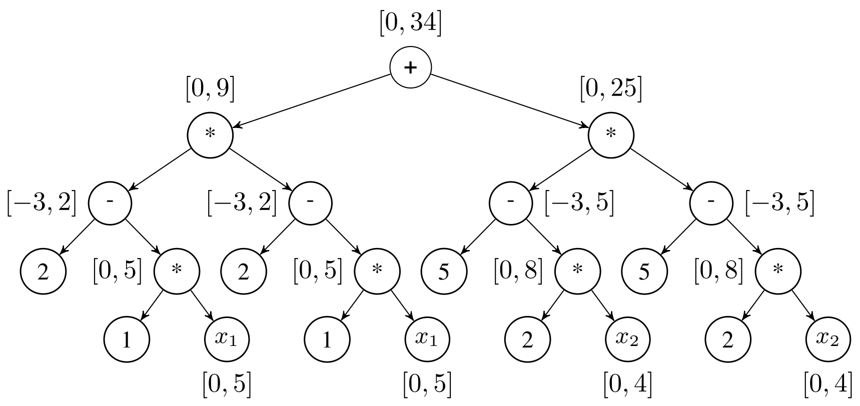


Figure 2.5: First Expression Tree Forward Propagation for $(2 - x_1)^2 + (5 - 2x_2)^2 \leq 8$.

Backward Propagation. In contrast to the forward propagation, the backward propagation starts from the root node and works its way down. The propagation reasons about each triple of variables, a node with two children and the connecting expression. Looking at the triple at the root node, in order to update the bounds of the left child, i.e., on the expression $(2 - x_1)^2$, while bounding it by 8, i.e., the right hand side of the ellipsoid constraint, the propagation algorithm also considers the minimum of the expression $(5 - 2x_2)^2$ from its domain. Note that we can consider the minimum of the expression as we are dealing with interval domains for the variables. In other words, in backward propagation we consider the bounds of the parent node in order to find the *min* and *max* values of the expression as the bounds of the two children nodes, while enforcing (generalized) arc consistency on the triple. Whenever the domains are changed, the propagation algorithm updates these changes through the connected constraints to prune the domains of the variables. The entire backward propagation reasoning and all the updated bounds are shown on Figure 2.6. Note that there might be multiple passes of the backward propagation in order to establish (generalized) arc consistency.

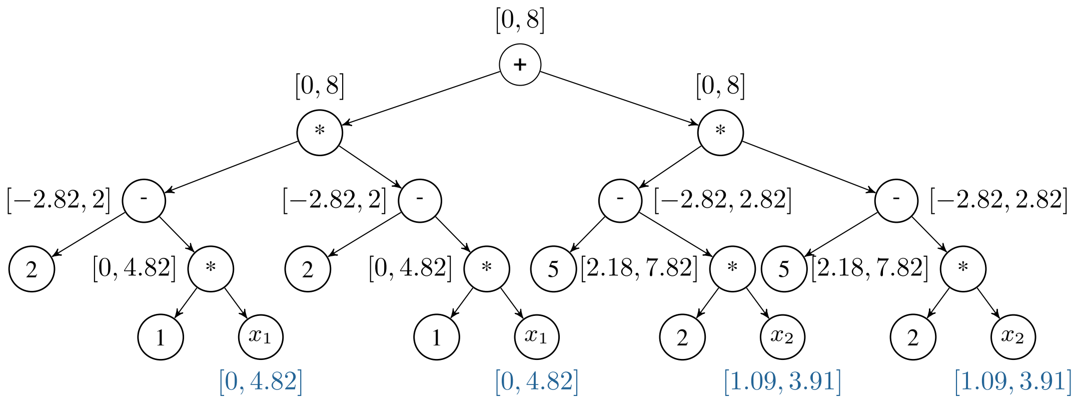


Figure 2.6: Final Expression Tree Backward Propagation for $(2-x_1)^2+(5-2x_2)^2 \leq 8$.

The obtained reduced bounds are $x_1 \in [0,4.82]$ and $x_2 \in [1.09,3.91]$. This example helps us to define the optimization problems that obtain the pruned minimum and maximum values for the variables which satisfy the given quadratic constraint and describe the constraint propagation algorithm.

Next, we generalize the approach and discuss the reasoning throughout the general expression tree for the ELLIPSOID constraint of the form $\sum_{i=1}^{n}(y_{ii}-a_{ii}x_i)^2 \leq \beta$ as shown on Figure 2.7. In Figure 2.7 we try to prune

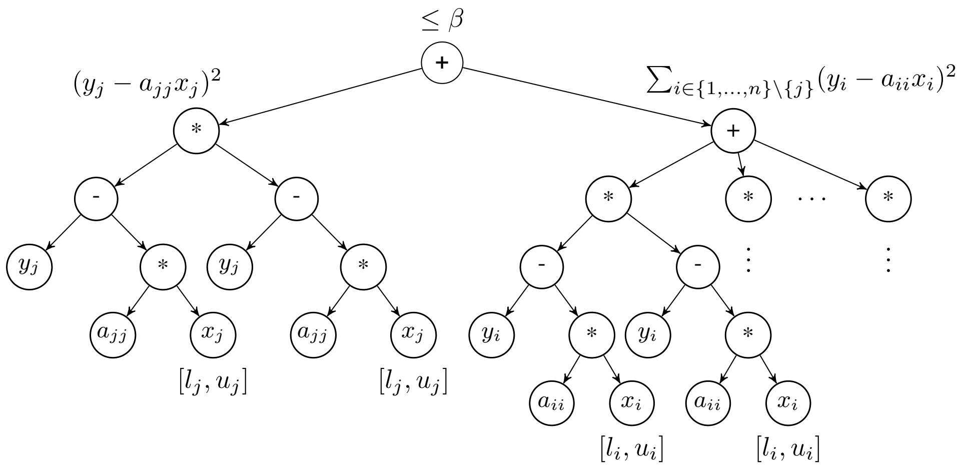


Figure 2.7: General Expression Tree for the ELLIPSOID constraint $\sum_{i=1}^{n}(y_i-a_{ii}x_i)^2 \leq \beta$.

the domains of the variable x_j where $N=\{1,\ldots,n\}$ represents the set of all variable indices. We partition the tree at the root node into two *main* branch sets, one containing the quadratic expression where x_j appears and the other for sum of the rest of the quadratic expressions. In order to prune the domain of x_j and find the tightest values for its bounds, we subtract the *minimum* of the expressions where x_j does not appear from the root node. Afterwards we enforce the generalized arc consistency on each triple of variables in the remaining subtree where x_j appears.

By understanding how the reasoning is done in the general expression tree, we define the optimization problems that CP_{DEFAULT} solves for the axis-aligned ellipsoid in order to obtain the pruned bounds for the variable x_j in the following manner:

$$l_j^{CP} := \min \ x_j \tag{2.16a}$$
$$\text{s.t.} \ (y_j - a_{jj}x_j)^2 \leq \beta - \sum_{i \in \{1,\ldots n\} \setminus \{j\}} \min_{l_i \leq x_i \leq u_i} (y_i - a_{ii}x_i)^2 \tag{2.16b}$$
$$l_j \leq x_j \leq u_j \tag{2.16c}$$

$$u_j^{CP} := \max \ x_j \tag{2.17a}$$
$$\text{s.t.} \ (y_j - a_{jj}x_j)^2 \leq \beta - \sum_{i \in \{1,\ldots n\} \setminus \{j\}} \min_{l_i \leq x_i \leq u_i} (y_i - a_{ii}x_i)^2 \tag{2.17b}$$
$$l_j \leq x_j \leq u_j \tag{2.17c}$$

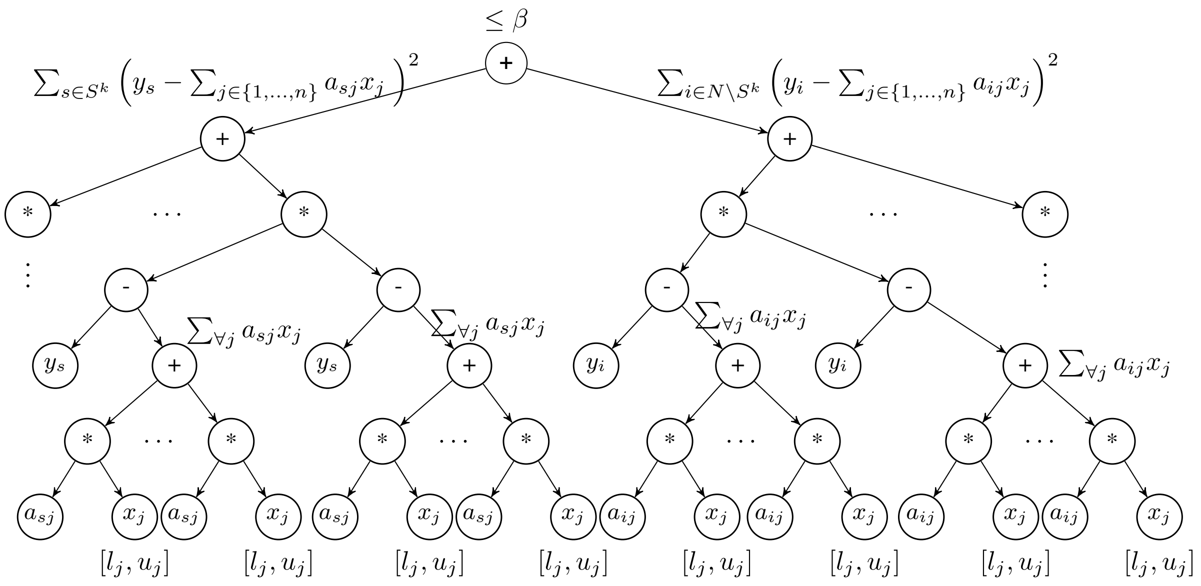
These two optimization problems are doing the exact same reasoning as the propagation algorithm. For example consider the problems for our example (2.15), for $j = 1$:

$$l_1^{CP} = \min \ x_1 \tag{2.18a}$$
$$\text{s.t.} \ (2 - x_1)^2 \leq 8 - \min_{0 \leq x_2 \leq 4} (5 - 2x_2)^2 \tag{2.18b}$$
$$0 \leq x_1 \leq 5 \tag{2.18c}$$

$$u_1^{CP} = \max \ x_1 \tag{2.19a}$$
$$\text{s.t.} \ (2 - x_1)^2 \leq 8 - \min_{0 \leq x_2 \leq 4} (5 - 2x_2)^2 \tag{2.19b}$$
$$0 \leq x_1 \leq 5 \tag{2.19c}$$

The minimum value that we can obtain for $(5 - 2x_2)^2$ with respect to x_2 bounds (2.18b, 2.19b) is zero. Following that statement, the constraints (2.18b) and (2.19b) are now reduced to $(2 - x_1)^2 \leq 8$. The solutions for this quadratic constraint when equality holds are $[2 - 2\sqrt{2}, 2 + 2\sqrt{2}] \Rightarrow [-0.82, 4.82]$. However we also consider the bounds for x_1 in (2.18c) and (2.19c). Once we intersect $[-0.82, 4.82]$ with x_1 bounds, we obtain the pruned domain for x_1 which is $[0, 4.82]$ (see Figure 2.6). We can obtain the pruned bounds for x_2 in a similar way.

CP_{DEFAULT} for Rotated Ellipsoid. The CP_{DEFAULT} propagation for rotated ellipsoids (2.14) reasons about the variable domains again by using expression tree propagation. We present in Figure 2.8 the general expression tree for the ELLIPSOID constraint for rotated ellipsoids. We denote with x_k the variable whose domain we want to update, while $N = \{1, \dots, n\}$ is the set of all variables indices. We also define $S^k = \{i \in \{1, \dots, n\} : a_{ik} \neq 0\}$, which represents the index set of quadratic expressions where the variable x_k appears. In the expression tree in Figure 2.8, we distinguish two *main* branch sets at the root node, one where x_k appears and the other one where x_k is not present. In order to obtain the best domain for x_k we subtract the sum of the *minimum* of the expressions where x_k does not appear from the root node and reduce the tree. Afterwards we propagate in the remaining subtree, enforcing generalized arc consistency on each triple of variables, as described in the previous paragraphs.



$\leq \beta$
$\sum_{s\in S^k} \left(y_s - \sum_{j\in\{1,\dots,n\}} a_{sj}x_j\right)^2$
$\sum_{i\in N\setminus S^k} \left(y_i - \sum_{j\in\{1,\dots,n\}} a_{ij}x_j\right)^2$
$\sum_{\forall j} a_{sj}x_j$
$\sum_{\forall j} a_{ij}x_j$
y_s
y_i
a_{sj}
a_{ij}
x_j
$[l_j, u_j]$

Figure 2.8: General Expression Tree for the ELLIPSOID constraint $\sum_{i=1}^{n}(y_i - \sum_{j=1}^{n} a_{ij}x_j)^2 \leq \beta$.

By understanding how CP$_{\text{DEFAULT}}$ conducts the reasoning in the expression tree, we can define the optimization problems that CP$_{\text{DEFAULT}}$ solves for reduced ellipsoids in order to obtain the pruned bounds for the variable x_k in the following manner:

$$l_k^{CP} := \min_{x\in\mathbb{R}^n} \quad x_k \tag{2.20a}$$

$$\text{s.t.} \quad \left(y_s - \sum_{j\in N} a_{sj}x_j\right)^2 \leq \beta - \sum_{i\in N\setminus S^k} \min_{\substack{l_j \leq x_j \leq u_j \\ j\in N}} \left(y_i - \sum_{j\in J} a_{ij}x_j\right)^2, \forall s \in S^k \tag{2.20b}$$

$$l_i \leq x_i \leq u_i, \forall i \in N \tag{2.20c}$$

$$
\begin{aligned}
u_k^{CP} := \max_{x \in \mathbb{R}^n} \quad & x_k && (2.21a)\\
\text{s.t.} \quad & \left(y_s - \sum_{j \in N} a_{sj} x_j \right)^2 \leq \beta - \sum_{i \in N \setminus S^k} \min_{\substack{l_j \leq x_j \leq u_j \\ j \in N}} \left(y_i - \sum_{j \in N} a_{ij} x_j \right)^2, \forall s \in S^k && (2.21b)\\
& l_i \leq x_i \leq u_i, \forall i \in N && (2.21c)
\end{aligned}
$$

2.6 Summary

In this chapter, we first presented an overview of the most well-known classes of optimization problems, namely LPs, QPs, QCQPs, SOCPs and SDPs. Then, we defined in more details the IQCPs. We also surveyed the literature of solution techniques for IQCPs, focusing on constraint programming as a fundamental technique used throughout this thesis. We explained the concept of a global constraint and then elaborated on the details of the ELLIPSOID global constraint. Lastly, we presented two propagation algorithms used for solving ellipsoid constrained problems, namely the BOX propagation algorithm and $\text{CP}_{\text{DEFAULT}}$ propagation algorithm.

In the rest of the thesis, we formally analyze these constraint propagation algorithms and gain valuable insights on when a propagation algorithm is dominant. In the following chapter we present the details of our analysis.

Chapter 3

An Analysis of Constraint Propagation Algorithms for the Ellipsoid Constraint

In the previous chapter we reviewed the ELLIPSOID global constraint and the BOX propagation algorithm, as well as the $\text{CP}_{\text{Default}}$ propagation algorithm behind IBM ILOG CP Optimizer. In this chapter we focus on analyzing the strength of both propagation algorithms for the ELLIPSOID constraint. We are particularly interested in this analysis because since the ELLIPSOID global constraint was first introduced there has not been any formal analysis of the strength of the BOX propagation mechanism [32]. Therefore in this chapter we develop a formal propagation analysis and present detailed mathematical proofs for each of the cases that we investigate in order to develop a deeper understanding of when one propagation algorithm is dominant. With the analysis, we address the fact that by knowing the problem structure and the form of the ELLIPSOID constraint we can decide if the constraint is a good choice for solving a particular problem.

Contributions. We present a theoretical analysis of the strength of constraint propagation for the ELLIPSOID constraint. With this analysis, we identify some future applications of the ELLIPSOID constraint as well as future prospects of designing stronger propagation algorithms.

The chapter is organized as follows. In Section 3.1 we present an overview of the entire propagation analysis that is conducted in this chapter. In Section 3.2 we start with the analysis for axis-aligned hyper-ellipsoids, where we also focus on one special subcase (Section 3.2.1). In Section 3.3 we continue with the analysis of rotated hyper-ellipsoids where we also discuss some special subcases (Section 3.3.1). In Section 4.4 we discuss the main findings of the analysis. We end the chapter with some concluding remarks in Section 3.5.

3.1 Structure of the Analysis

In this section, we introduce the scope of the formal analysis that we present in this chapter. Figure 3.1 shows a graphical overview of the analysis. We begin by partitioning the ellipsoids into axis-aligned and rotated hyper-ellipsoids. In Figure 3.1 everything that is related to axis-aligned hyper-ellipsoids is represented in blue, while orange represents everything from the analysis related to rotated hyper-ellipsoids. When we have an axis-aligned hyper-ellipsoid, we identified only one interesting case where we can show equivalence between the propagation algorithms. For a rotated hyper-ellipsoid we first look at the bounds of the ellipsoid. In Section 3.3.1 we define the notions of *redundant* (Definition 3.3.1) and *narrow* bounds (Definitions 3.3.2, 3.3.3) that we use in this part of the analysis. We derive dominance for the case when we have redundant bounds. Afterwards we focus on the case where we have narrow bounds and investigate what happens if we assign at least one value to a variable in the hyper-ellipsoid and if we do not. We focus on the case of when we assign at least one value to a variable as we see that the main characterization in this case comes again from the type of variable bounds. Therefore we conclude the analysis by showing dominance for when we have different types of bounds, redundant and narrow, once we assign at least one value to a variable.

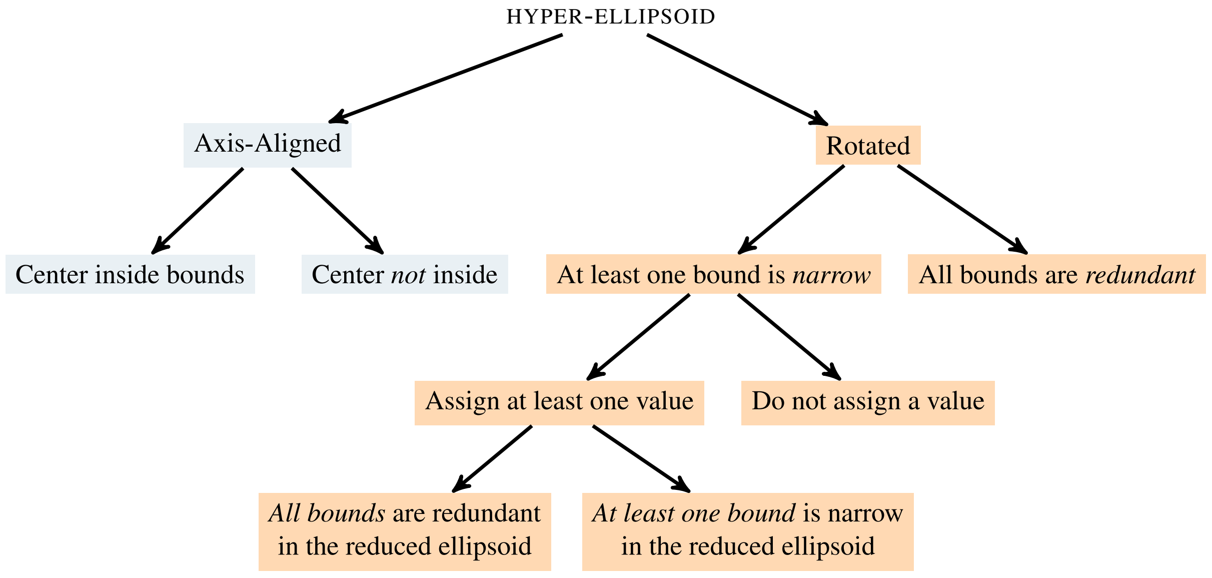

HYPER-ELLIPSOID
Axis-Aligned
Rotated
Center inside bounds
Center *not* inside
At least one bound is *narrow*
All bounds are *redundant*
Assign at least one value
Do not assign a value
All bounds are redundant in the reduced ellipsoid
At least one bound is narrow in the reduced ellipsoid

Figure 3.1: Overview of the Entire Formal Propagation Analysis for the ELLIPSOID constraint.

3.1.1 Preliminaries

For completeness, we repeat the basic notions and definitions regarding the propagation algorithms that were introduced in Chapter 2 as we frequently use them throughout this chapter. We also define the notation used for the comparison of two propagation algorithms performed throughout the analysis as well as the definition of a continuous as well as integer support of variables.

In Chapter 2 (Section 2.5) we defined the following optimization problems for CP$_{\text{DEFAULT}}$ propagation algorithm, both for axis-aligned and rotated ellipsoid as follows:

Axis-Aligned Ellipsoid

$$\begin{aligned} l_j^{CP} = \min\ & x_j \\ \text{s.t.}\ & (y_j - a_{jj}x_j)^2 \le \beta - \sum_{i\in\{1,\dots n\}\setminus\{j\}} \min_{l_i \le x_i \le u_i} (y_i - a_{ii}x_i)^2 \\ & l_j \le x_j \le u_j \end{aligned} \tag{3.1}$$

$$u_j^{CP} = \max\ x_j \tag{3.2a}$$

$$\text{s.t.}\ (y_j - a_{jj}x_j)^2 \le \beta - \sum_{i\in\{1,\dots n\}\setminus\{j\}} \min_{l_i \le x_i \le u_i} (y_i - a_{ii}x_i)^2 \tag{3.2b}$$

$$l_j \le x_j \le u_j \tag{3.2c}$$

Rotated Ellipsoid

$$\begin{aligned} l_k^{CP} = \min_{x\in\mathbb{R}^n}\ & x_k \\ \text{s.t.}\ & \left(y_s - \sum_{j\in N} a_{sj}x_j\right)^2 \le \beta - \sum_{i\in N\setminus S^k} \min_{\substack{l_j \le x_j \le u_j \\ j\in N}} \left(y_i - \sum_{j\in N} a_{ij}x_j\right)^2, \forall s \in S^k \\ & l_i \le x_i \le u_i, \forall i \in N \end{aligned} \tag{3.3}$$

$$u_k^{CP} = \max_{x\in\mathbb{R}^n}\ x_k \tag{3.4a}$$

$$\text{s.t.}\ \left(y_s - \sum_{j\in N} a_{sj}x_j\right)^2 \le \beta - \sum_{i\in N\setminus S^k} \min_{\substack{l_j \le x_j \le u_j \\ j\in N}} \left(y_i - \sum_{j\in N} a_{ij}x_j\right)^2, \forall s \in S^k \tag{3.4b}$$

$$l_i \le x_i \le u_i, \forall i \in N \tag{3.4c}$$

We also presented the following optimization problems that the BOX propagation algorithm solves in order to obtain the reduced bounds, both for axis-aligned and rotated ellipsoid:

Axis-Aligned Ellipsoid

$$\begin{aligned} l_j^{B} = \min_{x\in\mathbb{R}^n}\ & x_j \\ \text{s.t.}\ & \sum_{i=1}^{n} (y_i - a_{ii}x_i)^2 \le \beta \end{aligned} \tag{3.5}$$

$$\begin{aligned} u_j^B = \max_{x\in\mathbb{R}^n} \;& x_j \\ \text{s.t. } & \sum_{i=1}^{n}(y_i - a_{ii}x_i)^2 \leq \beta \end{aligned} \tag{3.6}$$

Rotated Ellipsoid

$$\begin{aligned} l_p^B = \min_{x\in\mathbb{R}^n} \;& x_p \\ \text{s.t. } & \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2 \leq \beta \end{aligned} \tag{3.7}$$

$$\begin{aligned} u_p^B = \max_{x\in\mathbb{R}^n} \;& x_p \\ \text{s.t. } & \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{n} a_{ij}x_j\right)^2 \leq \beta \end{aligned} \tag{3.8}$$

In terms of the structure of the analysis, we present fundamental definitions that we use for comparing two propagation algorithms. Given two propagation algorithms p_1 and p_2 we denote with D the vector of the domains, i.e., $D = [D_0, D_1, \ldots, D_n]$ where $D_i, i \in \{1, \ldots, n\}$ is the domain of variable x_i. We denote by $p(D)$ the vector of domains we obtain after running the propagation algorithm p on D and we denote by $p(D_i)$ the obtained domain of variable x_i after running propagation p. We denote the relation $D_i \subseteq E_i, \forall i \in \{1, \ldots, n\}$ by $D \preceq E$, and $(D \preceq E) \wedge (D_i \subset E_i), \exists i \in \{1, \ldots, n\}$ as $D \prec E$. We also denote the relation $D_i = E_i, \forall i \in \{1, \ldots, n\}$ as $D = E$.

Definition 3.1.1. A propagation algorithm p_1 is *at least as good as* (or not worse than) the propagation algorithm p_2 *iff:*

$$p_1(D) \preceq p_2(D)$$

We denote this relationship with $p_1 \succeq p_2$.

Definition 3.1.2. A propagation algorithm p_1 is *strictly better than* the propagation algorithm p_2 *iff*

$$p_1(D) \prec p_2(D)$$

We denote this relationship with $p_1 \succ p_2$.

Definition 3.1.3. Two propagation algorithms p_1 and p_2 are *incomparable iff* $\exists\, i, j \in \{1, \dots, n\}$ such that:

$$p_1(D_i) \npreceq p_2(D_i) \wedge p_2(D_j) \npreceq p_1(D_j)$$

Definition 3.1.4. Two propagation algorithms p_1 and p_2 are *equivalent iff* $p_1(D) = p_2(D)$. We denote this relationship with $p_1 \equiv p_2$.

Throughout our analysis, we first provide all the proofs for when we have a *continuous support* of variables, and then discuss the same cases for when we have *integer support* of variables in Section 3.2.2 and Section 3.3.2, respectively. We define continuous support of variables in the following way.

Let x_j be a domain variable and $D^{\mathbb{R}}(x_j) = [l_j, u_j]$ be the *continuous* interval domain of values that can be assigned to x_j.

Definition 3.1.5 (Continuous support)**.** For each variable $x_j \in \mathbb{R},\ j \in \{1, \dots n\}$ and every value $v_j \in D^{\mathbb{R}}(x_j)$ there exist values $v_i \in D^{\mathbb{R}}(x_i), \forall i \in \{1, \dots n\}, i \neq j$ such that the ELLIPSOID$(\{x_1 = v_1, \dots x_n = v_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta)$ is satisfied.

Next, we define integer support of variables in the following way. Let $D^{\mathbb{Z}}(x_j) = [l_j, u_j]$ be the *integer* interval domain of values that can be assigned to x_j, i.e, $\{l_j, l_j + 1, \dots, u_j\}$ where $l_j, u_j \in \mathbb{Z}$.

Definition 3.1.6 (Integer support)**.** For each variable $x_j \in \mathbb{Z},\ j \in \{1, \dots n\}$ and every value $v_j \in D^{\mathbb{Z}}(x_j)$ there exist values $v_i \in D^{\mathbb{Z}}(x_i), \forall i \neq j$ such that the ELLIPSOID$(\{x_1 = v_1, \dots x_n = v_n\}, \boldsymbol{A}, \boldsymbol{y}, \beta)$ is satisfied.

3.2 Axis-Aligned Hyper-Ellipsoid

In this section we focus on axis-aligned hyper-ellipsoids (Figure 3.2). We show that the CP$_{\text{DEFAULT}}$ propagation algorithm is at least as good as the BOX propagation algorithm and can be strictly better (Theorem 3.2.3).

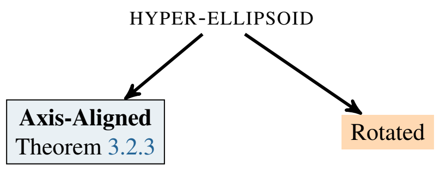


Figure 3.2: The outlined blue rectangle with bold text shows the next case to be explored.

Lemma 3.2.1. *For axis-aligned hyper-ellipsoids, CP*$_{\text{DEFAULT}}$ $\succeq$ *BOX propagation.*

Proof. Let x_j be an arbitrary variable with domain $[l_j, u_j]$. Consider the domains $[l_j^{CP}, u_j^{CP}]$ and $[l_j^{BOX}, u_j^{BOX}]$ obtained after applying CP$_{\text{DEFAULT}}$ and BOX propagation algorithms, respectively. We first show that $u_j^{CP} \leq u_j^{BOX}$.

Consider the optimization problems for CP$_{\text{DEFAULT}}$ and BOX propagation algorithm defined with (3.2) and (3.6), respectively. We will show that (3.6) is a relaxation of (3.2). Let $\tilde{x}_j$ be a feasible solution to (3.2). Also, let

$\tilde{x}_i$ be an optimal solution to the minimization problem in the right-hand-side of (3.2b). Then, $\tilde{x}$ is feasible to (3.6) as we have:

$$(y_j - a_{jj}\tilde{x}_j)^2 \leq \beta - \sum_{i \in \{1,\dots n\} \setminus \{j\}} (y_i - a_{ii}\tilde{x}_i)^2 \Rightarrow \sum_{i=1}^{n} (y_i - a_{ii}\tilde{x}_i)^2 \leq \beta \tag{3.9}$$

Note that the objective functions of (3.2) and (3.6) are the same.

Therefore, we obtain $u_j^{CP} \leq u_j^B$. We also know that $u_j^{CP} \leq u_j$ due to (3.2c), so we get $u_j^{CP} \leq \min\{u_j, u_j^B\} = u_j^{BOX}$.

Using similar arguments, it is easy to show that (3.5) is a relaxation of (3.1), thus $l_j^{CP} \geq l_j^{BOX}$, which concludes the proof. □

Lemma 3.2.2. *There exist axis-aligned hyper-ellipsoids and their respective variable domains where CP*$_{\text{DEFAULT}} \succ$ *BOX propagation.*

Proof. We consider the following ellipsoid constraint, that matches the form of the general ellipsoid constraint for axis-aligned hyper-ellipsoids (2.11), and its respective variable domains:

$$(12 - 3x_1)^2 + (18 - 6x_2)^2 \leq 144, \qquad x_1 \in [-1, 9],\ x_2 \in [0, 2]. \tag{3.10}$$

From the ellipsoid constraint we have

$$y = \begin{bmatrix} 12 \\ 18 \end{bmatrix},\ A^{-1} = \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{6} \end{bmatrix}$$

For x_1 we can calculate l_1^B and u_1^B , using (2.12) and (2.13) respectively as :

$$l_1^B = \frac{\sqrt{144}}{3} - \frac{12}{3} = \frac{12}{3} - \frac{12}{3} = 4 - 4 = 0$$
$$u_1^B = \frac{\sqrt{144}}{3} + \frac{12}{3} = \frac{12}{3} + \frac{12}{3} = 4 + 4 = 8$$

thus $l_1^{BOX} = \max\{-1, 0\} = 0$ and $u_1^{BOX} = \min\{9, 8\} = 8$.

Similarly for x_2 we have the tangent points calculated by BOX using (2.12) and (2.13):

$$l_2^B = 1$$
$$u_2^B = 5$$

thus $l_2^{BOX} = \max\{0, 1\} = 1$ and $u_2^{BOX} = \min\{2, 5\} = 2$. Then the resulting reduced bounds from BOX algorithm are $x_1 \in [0, 8]$ and $x_2 \in [1, 2]$.

However, CP$_{\text{DEFAULT}}$ reduces the bounds to $x_1 \in [0.53, 7.46]$ and $x_2 \in [1, 2]$. The expression trees for forward and backward propagation are given in Figure 3.3 and Figure 3.4, respectively.

□

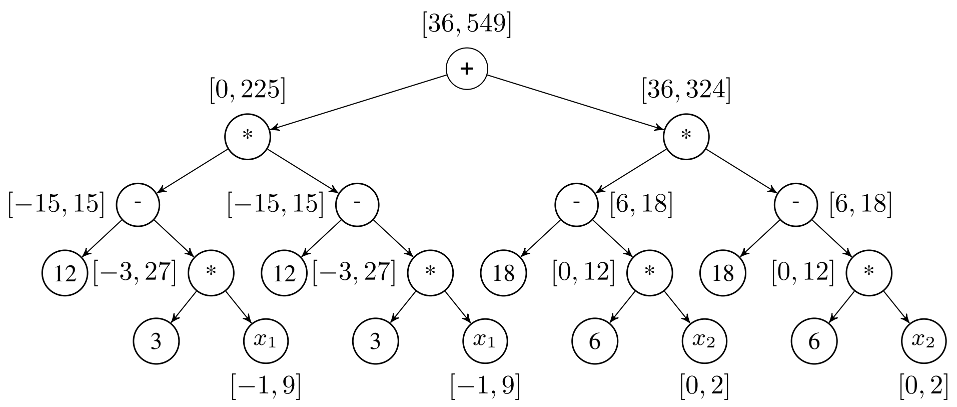


Figure 3.3: First Expression Tree (Forward) Propagation for $(12-3x_1)^2+(18-6x_2)^2 \leq 144, x_1 \in [-1,9],\ x_2 \in [0,2]$.

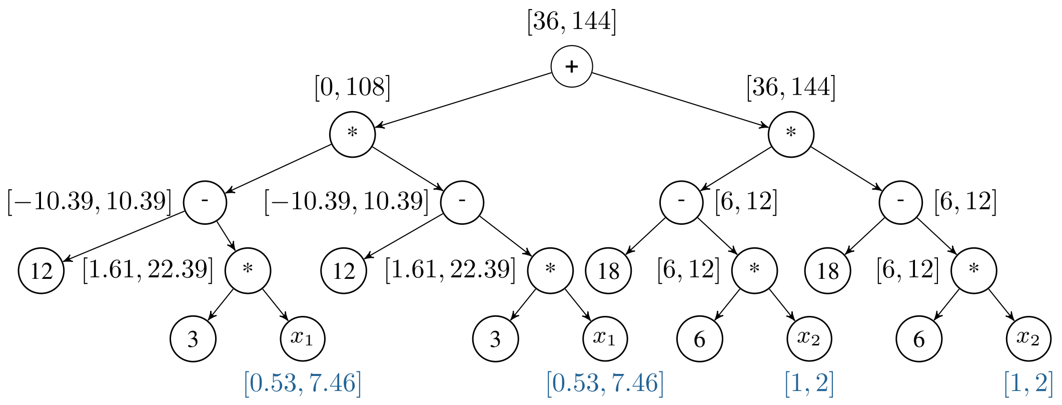


Figure 3.4: Final Expression Tree (Backward) Propagation for $(12-3x_1)^2+(18-6x_2)^2 \leq 144, x_1 \in [-1,9],\ x_2 \in [0,2]$. The final reduced bounds are highlighted.

Theorem 3.2.3. *For axis-aligned hyper-ellipsoids CP*$_{\text{DEFAULT}} \succ$ *BOX propagation.*

Proof. Follows directly from Lemma 3.2.1 and Lemma 3.2.2. □

3.2.1 Special Subcase

In this section we delve into more details of when one propagation algorithm is dominant. We identify one special case where we prove that the CP$_{\text{DEFAULT}}$ propagation and the BOX propagation are equivalent, and note an implication that follows from it.

We first define a specific hypercube to be used in some of the proofs.

Definition 3.2.1. A *domain hypercube* is defined as the Cartesian product of the interval domains of the variables $\times_{j=1}^{n}\ [l_j, u_j]$.

We formally prove our claim that when the hypercube defined by the variable bounds includes the center of the ellipsoid, the CP$_{\text{DEFAULT}}$ propagation algorithm and the BOX propagation algorithm are equivalent.

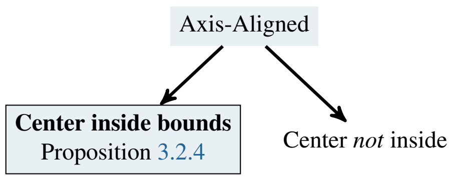


Figure 3.5: The outlined blue rectangle with bold text shows the next case to be explored.

Proposition 3.2.4. *If the domain hypercube includes the center of the ellipsoid then CP*$_{\text{DEFAULT}}$ $\equiv$ *BOX propagation algorithm for axis-aligned hyper-ellipsoids.*

Proof. Let x_j be an arbitrary variable with domain $[l_j, u_j]$. Consider the domains $[l_j^{CP}, u_j^{CP}]$ and $[l_j^{BOX}, u_j^{BOX}]$ obtained after applying CP$_{\text{DEFAULT}}$ and BOX propagation algorithms, respectively. Also suppose that the center of the ellipsoid belongs to the domain hypercube. We first show that $u_j^{CP} = u_j^{BOX}$.

Consider the optimization problems defined by (3.2) and (3.6). We analyze the optimal solutions for both of the problems. Since we aim at maximizing x_j we need to maximize the right-hand-side of both (3.2) and (3.6). As (3.6) is defined over $\mathbb{R}^n$, the minimum value that we can obtain for the sum of the quadratic expressions subtracted from β, attained at $x_i = \frac{y_i}{a_{ii}}, \forall i \in \{1,\dots,n\}\backslash\{j\}$, is 0. So for (3.6) we get the following form:

$$u_j^B = \max_{x_j \in \mathbb{R}} x_j \quad \text{s.t.} \quad (y_j - a_{jj}x_j)^2 \leq \beta$$

Thus for the optimal solution of (3.6) we get:

$$u_j^B = \frac{y_j}{a_{jj}} + \left|\frac{\sqrt{\beta}}{a_{jj}}\right|$$

Next, let $\delta_j := \sum_{i\in\{1,\dots,n\}\backslash\{j\}} \min_{l_i \leq x_i \leq u_i} (y_i - a_{ii}x_i)^2$. According to our assumption that the center of the ellipsoid belongs to the domain hypercube, i.e., $\frac{y_i}{a_{ii}} \in [l_i, u_i]$ it follows that $\delta_j = 0$. That is, (3.2) is equivalent to :

$$\begin{aligned} u_j^{CP} = \max_{l_j \leq x_j \leq u_j} \ & x_j \\ \text{s.t. } & (y_j - a_{jj}x_j)^2 \leq \beta \end{aligned}$$

Then for the optimal solution we get:

$$\begin{aligned} u_j^{CP} &= \min\left\{u_j, \frac{y_j}{a_{jj}} + \left|\frac{\sqrt{\beta}}{a_{jj}}\right|\right\} \\ &= \min\{u_j, u_j^B\} = u_j^{BOX} \end{aligned}$$

Using similar arguments, it is easy to show that $l_j^{CP} = l_j^{BOX}$, which concludes the proof. □

Note than Proposition 3.2.4 implies that if CP$_{\text{DEFAULT}}$ propagation algorithm can do more inference than the BOX propagation algorithm, then the center does not belong to the domain hypercube.

3.2.2 Integer Support

In the previous section, we proved Theorem 3.2.3 and Proposition 3.2.4 only considering a *continuous support* for variables. However in constraint programming the most often used support is integer support for variables.

In this section, we present a general discussion on the changes in the analysis if we have an integer support of variables for the investigated cases of axis-aligned hyper-ellipsoid. The rest of the details, such as expression trees and detailed proofs are provided in Appendix A and Appendix B, respectively. As the set of integers is a subset of the set of real numbers most of our analysis holds true when we have integer support.

Starting with Lemma 3.2.1, the main argument that we are using to show that $\mathrm{CP}_{\mathrm{DEFAULT}} \succeq$ BOX propagation is that (3.6) is a relaxation of (3.2). With integer support, each variable domain is an integer interval set. By having integer bounds, problem (3.2) is even more constrained and the integer bounds are a subset of $\mathbb{Z}^n$ - where (3.6) is now defined. Thus, $\mathrm{CP}_{\mathrm{DEFAULT}} \succeq$ BOX propagation for when we have integer support and the proof holds true.

In Table 3.1 we give the details for the example in Lemma 3.2.2, but with integer support for the variables where the best bounds are highlighted. We see that we get the same result as in Lemma 3.2.2, which is that $\mathrm{CP}_{\mathrm{DEFAULT}} \succ$ BOX propagation algorithm.

Table 3.1: Bound comparison for $(12 - 3x_1)^2 + (18 - 6x_2)^2 \leq 144$ with *integer support* for variables. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
$\mathrm{CP}_{\mathrm{DEFAULT}}$	1	7	1	2
BOX	0	8	1	2

By knowing that Lemma 3.2.1 and Lemma 3.2.2 can be proved in the same manner for integer support and have the same conclusions, we can also conclude that Theorem 3.2.3 holds true for integer support as well.

Furthermore, coming to the proof of Proposition 3.2.4, note that the integer support is reflected in the integer bounds, while the parameters that describe the center of the ellipsoid do not have an integer restriction. In the proof of Proposition 3.2.4 we analyze the optimal solutions for the problems that $\mathrm{CP}_{\mathrm{DEFAULT}}$ and BOX propagation algorithms solve, hence we do not use integer support as an argument explicitly. Therefore we have the same conclusions for Proposition 3.2.4 for when we have integer support.

3.3 Rotated Hyper-Ellipsoid

In this section we compare the BOX propagation (3.7, 3.8) with the $\mathrm{CP}_{\mathrm{DEFAULT}}$ propagation algorithm (3.3,3.4) when we have a rotated hyper-ellipsoid. We first show that the two propagation algorithms are incomparable through two counterexamples.

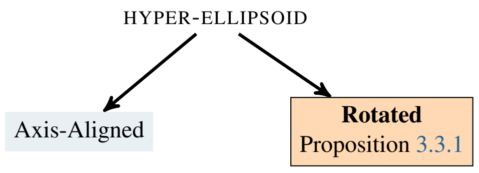


Figure 3.6: The outlined orange rectangle with bold text shows the next case to be explored.

Proposition 3.3.1. *The CP*$_{\text{DEFAULT}}$ *propagation and the BOX propagation are incomparable for rotated hyper-ellipsoids.*

To formally prove Proposition 3.3.1, we present two counterexamples which show that there does not exist a dominance between the BOX propagation and the CP$_{\text{DEFAULT}}$ propagation for rotated hyper-ellipsoids.

Counterexample 3.3.1 (BOX $\succ$ CP$_{\text{DEFAULT}}$)**.** First, we present an example where the BOX propagation does more inference than the CP$_{\text{DEFAULT}}$ propagation algorithm for rotated hyper-ellipsoids. We consider the following ellipsoid constraint, that matches the form of the general ellipsoid constraint for rotated hyper-ellipsoids (2.14):

$$(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196, \quad x_1 \in [-3,2],\ x_2 \in [-2,3],\ x_3 \in [2,2]$$

The CP$_{\text{DEFAULT}}$ propagation reduces the bounds for x_1 to [-2, 2] and the x_2 bounds to [-2, 2.5]. The forward and backward propagation are shown on Figure 3.7 and Figure 3.8, respectively.

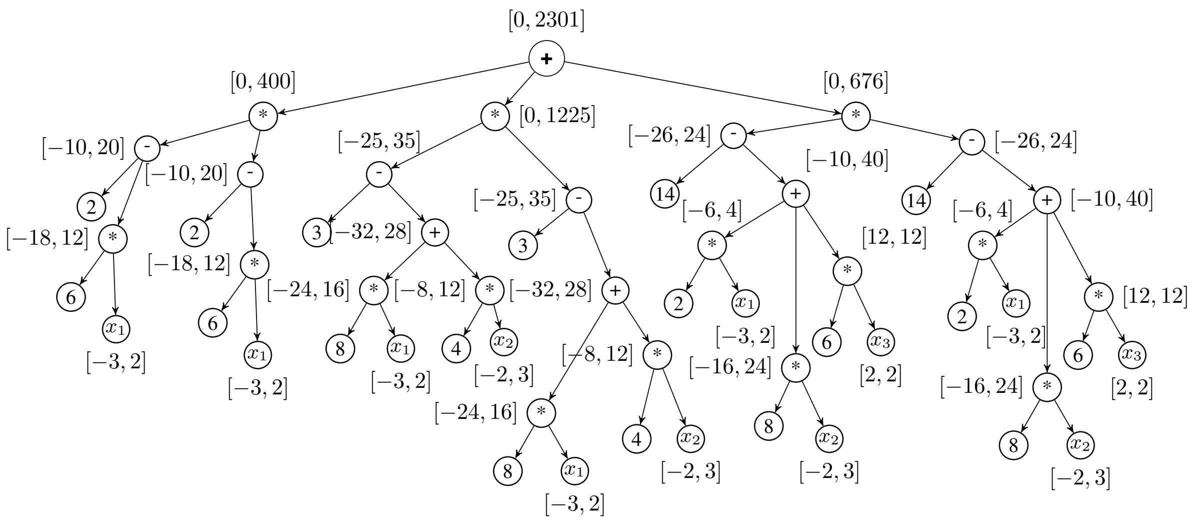


Figure 3.7: First Expression Tree (Forward) Propagation for the ELLIPSOID:$(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196, x_1 \in [-3,2],\ x_2 \in [-2,3],\ x_3 \in [2,2]$.

In order to show the detailed BOX propagation for this example, we first need to find and calculate the one-dimension-smaller (reduced) ellipsoid [32] as $x_3 \in [2,2]$. Let $\tilde{A} = [A_j]$ for all non-fixed variables, where A_j is

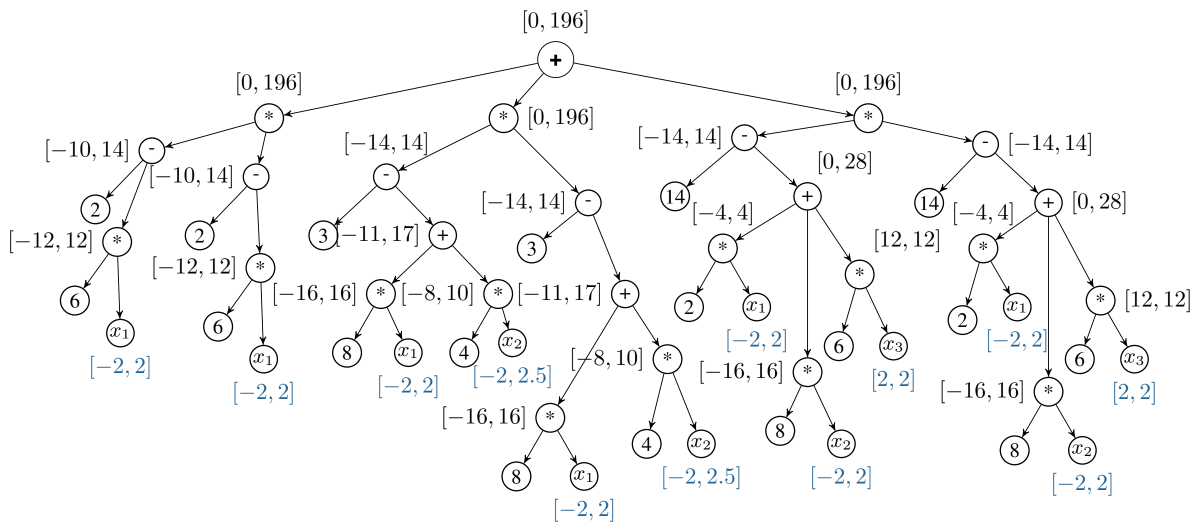


Figure 3.8: Final Expression Tree (Backward) Propagation for the ELLIPSOID:$(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196, x_1 \in [-3,2],\ x_2 \in [-2,3],\ x_3 \in [2,2]$. The final reduced bounds are highlighted.

the j^{th} column of A, so we have:

$$\tilde{A} = \begin{bmatrix} 6 & 0 \\ 8 & 4 \\ 2 & 8 \end{bmatrix}$$

We then calculate the QR factorization of $\tilde{A}$.

$$QR(\tilde{A}) = \left[\tilde{\boldsymbol{Q}}_1 | \tilde{\boldsymbol{Q}}_2\right] \begin{bmatrix} \tilde{\boldsymbol{R}} \\ 0 \end{bmatrix} = \left[\begin{array}{cc|c} -0.5883 & 0.3641 & 0.7220 \\ -0.7845 & -0.0405 & -0.6189 \\ -0.1961 & -0.9305 & 0.3094 \end{array}\right] \begin{bmatrix} -10.1980 & -4.7068 \\ 0 & -7.6057 \\ 0 & 0 \end{bmatrix}$$

We also calculate $\bar{y} = y - A_3 \times v_3$, where v_3 is the value assigned to x_3 as:

$$\bar{y} = \begin{bmatrix} 2 \\ 3 \\ 14 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}$$

and $\tilde{y} = \tilde{Q_1}^\top \bar{y}$:

$$\tilde{y} = \begin{bmatrix} -0.5883 & 0.3641 \\ -0.7845 & -0.0405 \\ -0.1961 & -0.9305 \end{bmatrix}^\top \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} -3.9223 \\ -1.2543 \end{bmatrix}$$

The reduced intersecting ellipsoid can be computed as $\varepsilon_F = \|\tilde{y} - \tilde{R}\tilde{x}\|_2^2 \leq \tilde{\beta}$, where for $\tilde{\beta}$ we have:

$$\tilde{\beta} = \beta - \|\bar{y}\|_2^2 + \|\tilde{y}\|_2^2 = 196 - 17 + 16.95 = 195.95$$

Finally the complete form of the reduced ellipsoid (Figure 3.9) is:

$$\varepsilon_F = \left\| \begin{bmatrix} \text{-3.9223} \\ \text{-1.2543} \end{bmatrix} - \begin{bmatrix} \text{-10.1980} & \text{-4.7068} \\ 0 & \text{-7.6057} \end{bmatrix} \tilde{x} \right\|_2^2 \leq 195.95$$

$$\Rightarrow (-3.9223 + (10.1980x_1 + 4.7068x_2))^2 + (-1.2543 + 7.6057x_2)^2 \leq 195.95 \tag{3.11}$$

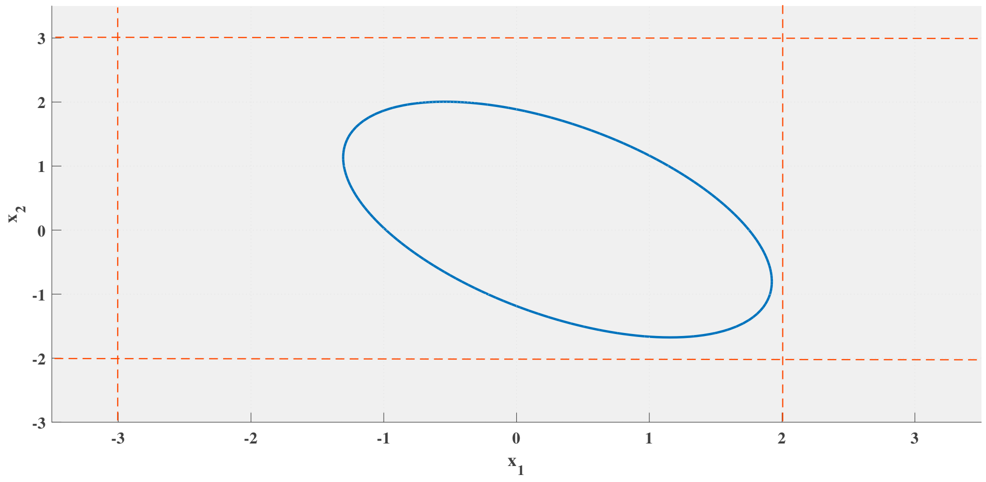


Figure 3.9: The Geometry of the Ellipsoid $(-3.9223 + (10.1980x_1 + 4.7068x_2))^2 + (-1.2543 + 7.6057x_2)^2 \leq 196, x_1 \in [-3, 2],\ x_2 \in [-2, 3],\ x_3 \in [2, 2]$.

Now, knowing the reduced ellipsoid we can calculate the tangent box. Since (3.11) is written in the form of the general ellipsoid constraint for rotated hyper-ellipsoids we can identify:

$$y = \begin{bmatrix} \text{-3.9223} \\ \text{-1.2543} \end{bmatrix}, A = \begin{bmatrix} \text{-10.1980} & \text{-4.7068} \\ 0 & \text{-7.6057} \end{bmatrix} \text{ and } A^{-1} = \begin{bmatrix} \text{-0.0981} & 0.0607 \\ 0 & \text{-0.1315} \end{bmatrix}$$

We then calculate l_j^B and $u_j^B, \forall j = \{1, 2\}$ using (2.12) and (2.13)

$$l_1^B = 0.3085 - 1.6151 = -1.3066$$

$$u_1^B = 0.3085 + 1.6151 = 1.9236$$

$$l_2^B = 0.1649 - 1.8410 = -1.6761$$
$$u_2^B = 0.1649 + 1.8410 = 2.0059$$

The resulting reduced bounds from the BOX algorithm are $x_1 \in [-1.3066, 1.9236]$ and $x_2 \in [-1.6761, 2.0059]$

Thus, the BOX propagation reduces the bounds more than the CP$_{\text{DEFAULT}}$ propagation, so it does more inference in this example.

Counterexample 3.3.2 (CP$_{\text{DEFAULT}} \succ$ BOX)**.** Next, we present an example which shows that the CP$_{\text{DEFAULT}}$ propagation can do more propagation than the BOX propagation for rotated hyper-ellipsoids. We consider the following ellipsoid constraint (Figure 3.10), that matches the form of the general ellipsoid constraint for rotated hyper-ellipsoids (2.14):

$$(2 - x_1)^2 + (5 - (x_1 + x_2))^2 \leq 8, \qquad x_1 \in [-1, 5],\; x_2 \in [5, 7]$$

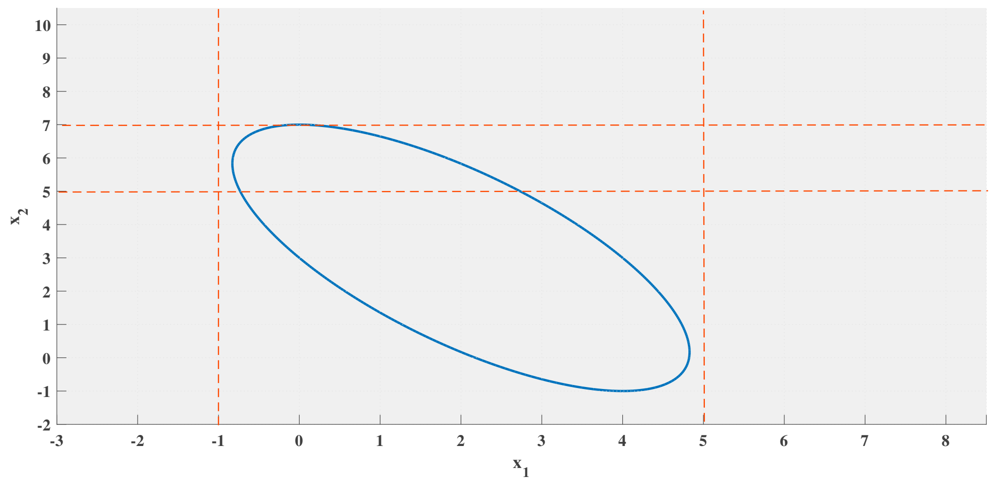


Figure 3.10: Geometry of the Ellipsoid $(2 - x_1)^2 + (5 - (x_1 + x_2))^2 \leq 8, x_1 \in [-1, 5],\; x_2 \in [5, 7]$.

From the constraint we know:

$$y = \begin{bmatrix} 2 \\ 5 \end{bmatrix}, A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } A^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

We can calculate the lower and upper bounds as solutions to (2.12) and (2.13):

$$l_1^B = -0.827 \qquad l_2^B = -1$$
$$u_1^B = 4.823 \qquad u_2^B = 7$$

Upon intersection with the starting bounds, the resulting reduced bounds from the BOX propagation are

$x_1 \in [-0.82, 4.82], x_2 \in [5, 7]$. However CP$_{\text{DEFAULT}}$ reduces more the bounds to $x_1 \in [-0.82, 2.82]$ and keeps $x_2 \in [5, 7]$. The forward and backward propagation that lead to these bounds are shown on Figure 3.11 and Figure 3.12, respectively.

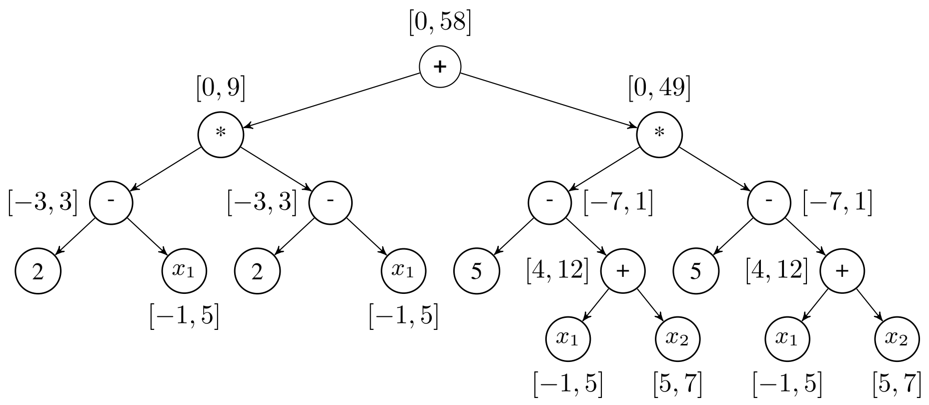


Figure 3.11: First Expression Tree (Forward) Propagation for $(2-x_1)^2+(5-(x_1+x_2))^2 \leq 8, x_1 \in [-1, 5],\ x_2 \in [5, 7]$.

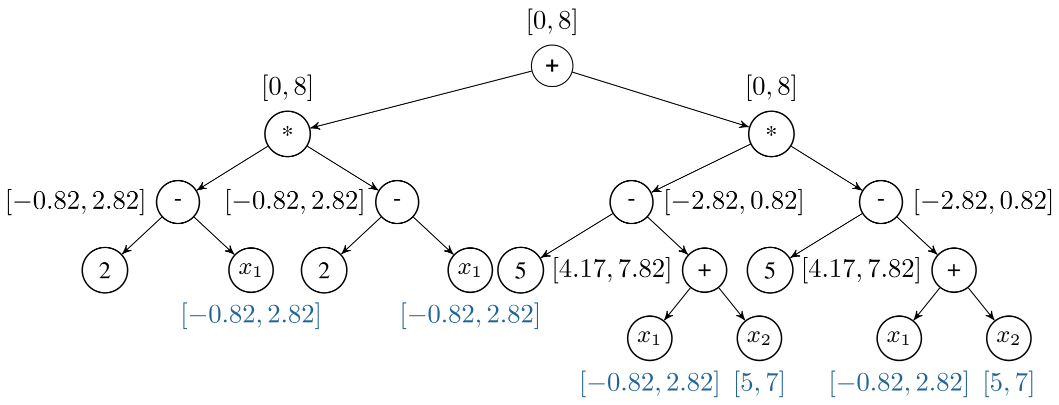


Figure 3.12: Final Expression Tree (Backward) Propagation for $(2 - x_1)^2 + (5 - (x_1 + x_2))^2 \leq 8, x_1 \in [-1, 5],\ x_2 \in [5, 7]$. The final reduced bounds are highlighted.

3.3.1 Special Subcases

In this section we further investigate when one propagation algorithm is dominant in the case of a rotated hyper-ellipsoid. In order to present the first characterization, we need to define some preliminary notions. Recall that l_j^B, u_j^B denote the bounds of the tangent box of the rotated hyper-ellipsoid, as previously defined in Section 2.4.

Definition 3.3.1 (Redundant bounds)**.** Given l_j^B and u_j^B as the bounds of the tangent box of the hyper-ellipsoid for a particular variable x_j, its variable bounds l_j and u_j are said to be *redundant* if $l_j^B \geq l_j$ and $u_j^B \leq u_j$, respectively. An example of such redundant bounds is shown in Figure 3.13.

Definition 3.3.2 (Narrow lower bound)**.** Given l_j^B as the lower bound of the tangent box of the hyper-ellipsoid for a particular variable x_j, its variable lower bound l_j is said to be *narrow* if $l_j^B < l_j$. An example of such narrow lower bound is shown in Figure 3.14.

Definition 3.3.3 (Narrow upper bound)**.** Given u_j^B as the upper bound of the tangent box of the hyper-ellipsoid for a particular variable x_j, its variable upper bound u_j is said to be *narrow* if $u_j^B > u_j$. An example of such narrow upper bound is shown in Figure 3.14.

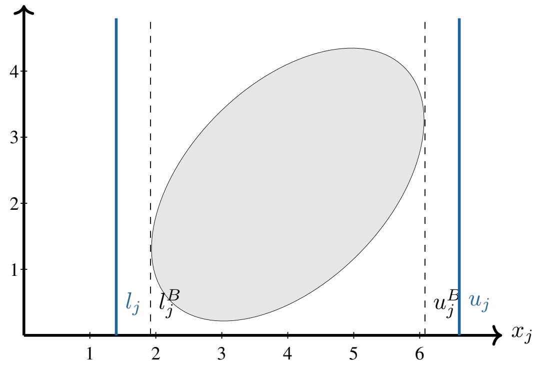


Figure 3.13: An example of redundant bounds. The dashed lines represent the tangent bound of the BOX, while the blue lines are the redundant bounds.

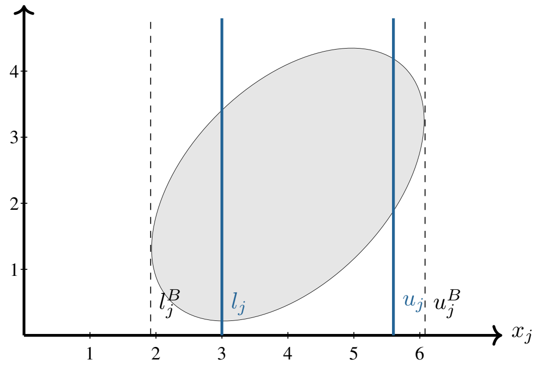


Figure 3.14: An example of narrow bounds. The dashed lines represent the tangent bound of the BOX, while the blue lines represent the narrow bounds.

Based on the presented definitions we identify two special cases where we compare the propagation algorithms. First, we formally prove that when all the variable bounds are redundant (Figure 3.15), then the BOX propagation is not worse than the CP$_{\text{DEFAULT}}$ propagation algorithm (Lemma 3.3.2) and can be strictly better (Lemma 3.3.3). In the first lemma, we prove by contradiction that the BOX propagation is not worse than the CP$_{\text{DEFAULT}}$ propagation, while in the second lemma we provide an example where the BOX propagation can do strictly better than the CP$_{\text{DEFAULT}}$ propagation.

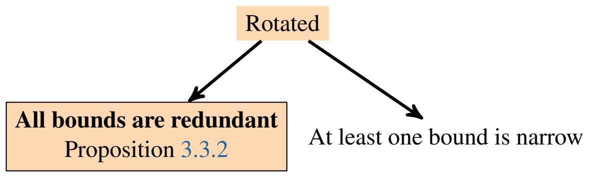


Figure 3.15: The outlined orange rectangle with bold text shows the next case to be explored.

Proposition 3.3.2. *If all variable bounds in the rotated hyper ellipsoid are redundant then BOX propagation* $\succeq$ *CP*$_{\text{DEFAULT}}$.

Proof. (By Contradiction.) Let u_j^{CP} denote the updated upper bound that CP$_{\text{DEFAULT}}$ obtains and u_j^{BOX} be the updated upper bound that BOX obtains. For a contradiction, without loss of generality, suppose that CP$_{\text{DEFAULT}}$

does more reduction on the upper bound of some variable $x_j, j \in \{1, \ldots, n\}$ than the BOX propagation and obtains u_j^{CP} where $u_j^{CP} < u_j^{BOX}$. Assuming that all variable bounds are redundant we know that $u_j^{CP} < u_j^B \leq u_j$ where u_j is the original upper bound of the variable x_j. Based on our assumption that the CP$_{\text{DEFAULT}}$ obtains a valid u_j^{CP} we know the problem (3.4) is feasible. This also implies that the problem (3.8) is feasible. Next, we consider the point x^B obtained as an optimal solution to the feasible BOX problem (3.8), where $x_j^B = u_j^B$. The point x^B satisfies all variable bounds as they are all redundant and x^B lies inside the ellipsoid. This point is also feasible to the CP$_{\text{DEFAULT}}$ problem (3.4) as it lies inside the ellipsoid, meaning it satisfies the ELLIPSOID constraint. Our initial assumption was that CP$_{\text{DEFAULT}}$ propagation does strictly better than the BOX, which means it cuts off this point. However the CP$_{\text{DEFAULT}}$ propagation is a sound propagation algorithm, meaning it cannot remove any feasible solution such as this point thus we have a contradiction. □

Proposition 3.3.3. *There exist rotated hyper-ellipsoids with redundant bounds where BOX propagation $\succ$ CP$_{\text{DEFAULT}}$.*

Proof. The proof follows directly from Counterexample 3.3.1 from Proposition 3.3.1 as the bounds of all the variables are redundant. The details regarding the example are shown in Table 3.5, where the best bounds found by BOX propagation algorithm are highlighted.

Table 3.2: Bound comparison for $(2 - 6x_1)^2 + (3 - (8x_1 + 4x_2))^2 + (14 - (2x_1 + 8x_2 + 6x_3))^2 \leq 196$. The best bounds are highlighted.

	x_1		x_2		x_3	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
Original	-3	2	-2	3	2	2
Tangent box	-1.33	1.92	-1.67	2	2	2
CP$_{\text{DEFAULT}}$	-2	2	-2	2.50	2	2
BOX	-1.33	1.92	-1.67	2	2	2

□

Next, we prove that when at least one of the variable bounds is narrow in the rotated hyper-ellipsoid (Figure 3.16) then the CP$_{\text{DEFAULT}}$ propagation and the BOX propagation are incomparable. To do so, we present two counter-examples which show that there does not exist a dominance between the BOX propagation and the CP$_{\text{DEFAULT}}$ propagation for this case.

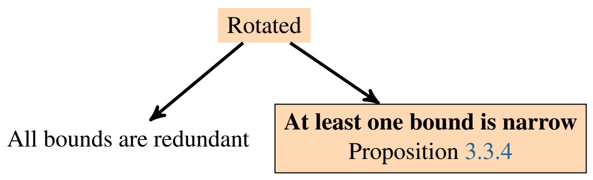


Figure 3.16: The outlined orange rectangle with bold text shows the next case to be explored.

Proposition 3.3.4. *The BOX propagation and the CP$_{\text{DEFAULT}}$ propagation are incomparable when at least one of the variable bounds is narrow.*

Counterexample 3.3.3 (CP$_{\text{DEFAULT}}$ $\succ$ BOX)**.** We use the counterexample 3.3.2 presented in Proposition 3.3.1 as the lower bound on the variable x_2 is narrow. This example shows that the CP$_{\text{DEFAULT}}$ propagation does more reduction on the x_1 variable upper bound than the BOX propagation algorithm. The details regarding the example are shown in Table 3.3, where the best bound found by CP$_{\text{DEFAULT}}$ is highlighted.

Table 3.3: Bound comparison for $(2-x_1)^2+(5-(x_1+x_2))^2 \leq 8$. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
Original	-1	5	5	7
Tangent box	-0.83	4.82	-1	7
CP$_{\text{DEFAULT}}$	-0.83	2.82	5	7
BOX	-0.83	4.82	5	7

Counterexample 3.3.4 (BOX $\succ$ CP$_{\text{DEFAULT}}$)**.** We use the ellipsoid constraint from Counterexample 3.3.1 presented in Proposition 3.3.1, but with a different set of initial variable bounds given in Table 3.4. The best bounds found are highlighted. The BOX propagation calculates the reduced bounds $x_1 \in [-1.3066, 1.9236], x_2 \in [-1.6761, 2.0059]$. Upon intersection with the original bounds, the final bounds are $x_1 \in [-1, 1.9236], x_2 \in [-1.6761, 2.0059]$.

As for CP$_{\text{DEFAULT}}$, it reduces the bounds for x_1 to $[-1, 2]$ and for x_2 to $[-2, 2.25]$, showing a weaker reduction in the variable bounds. The forward and backward propagation trees are shown on Figure 3.17 and Figure 3.18, respectively.

Table 3.4: Bound comparison for $(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196$. The best bounds are highlighted.

	x_1		x_2		x_3	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
Original	-1	2	-4	4	2	2
Tangent BOX	-1.33	1.92	-1.67	2	2	2
CP$_{\text{DEFAULT}}$	-1	2	-2	2.25	2	2
BOX	-1	1.92	-1.67	2	2	2

3.3.1.1 Value Assignment to Variables

In this section we focus on rotated hyper-ellipsoid constraints where there is at least one variable bound that is narrow. We further investigate what happens in the case when we assign a value to one of the variables. When a variable is fixed to a certain value during the search, e.g., $x_i = v_i$ then the dimension of the ellipsoid is reduced

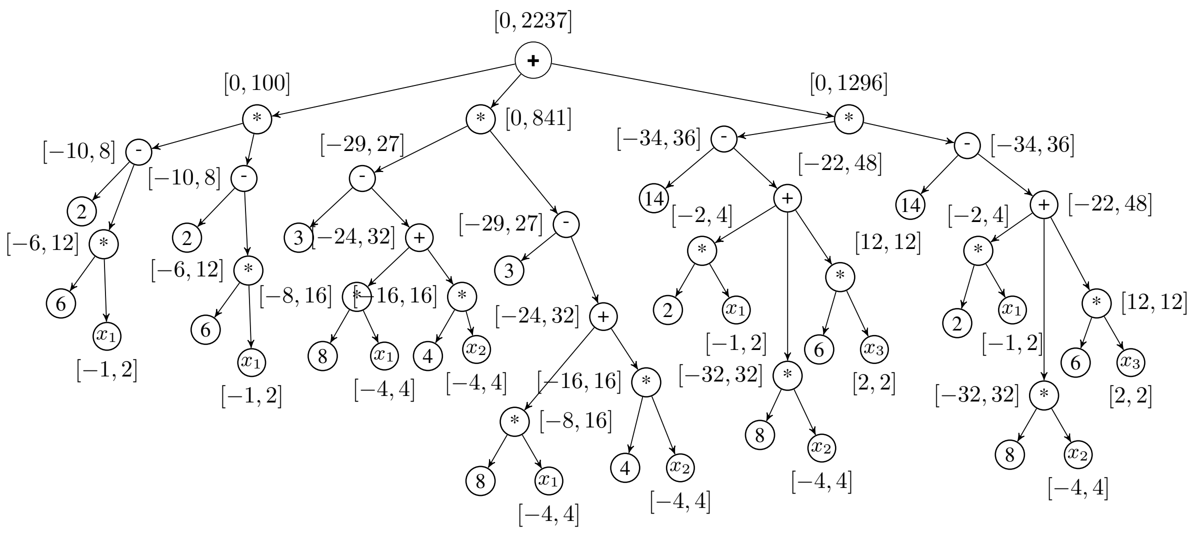


Figure 3.17: First Expression Tree (Forward) Propagation for the ELLIPSOID:$(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196, x_1 \in [-1,2],\ x_2 \in [-4,4],\ x_3 \in [2,2]$.

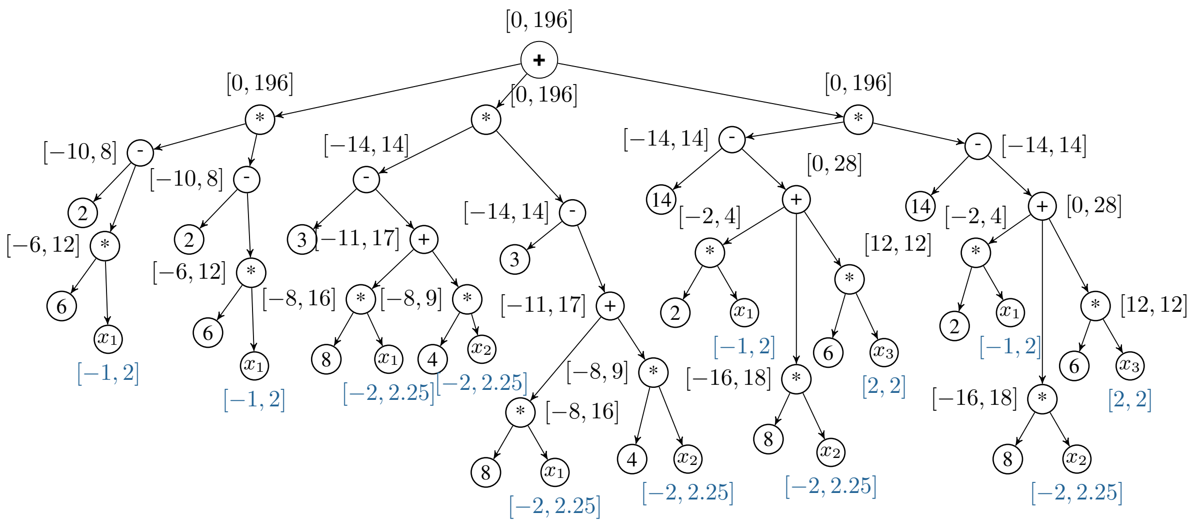


Figure 3.18: Final Expression Tree (Backward) Propagation for the ELLIPSOID:$(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196, x_1 \in [-1,2],\ x_2 \in [-4,4],\ x_3 \in [2,2]$. The final reduced bounds are highlighted.

by one [32]. After assigning a value to a variable there are two possible subcases: the resulting bounds after the assignment in the lower dimensional (reduced) ellipsoid either are all redundant or at least one of them is narrow.

First we formally prove that when we assign a value to one of the variables and we have redundant bounds in the lower dimensional ellipsoid (Figure 3.19) the BOX propagation is not worse than the $\text{CP}_{\text{Default}}$ propagation (Proposition 3.3.5).

Proposition 3.3.5. *If there is at least one fixed variable and all the bounds in the resulting reduced rotated hyper-ellipsoid are redundant, then BOX propagation $\succeq$ CP_{DEFAULT}.*

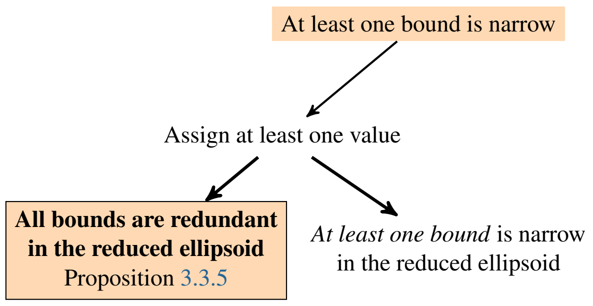

At least one bound is narrow
Assign at least one value
All bounds are redundant in the reduced ellipsoid
Proposition 3.3.5
At least one bound is narrow in the reduced ellipsoid

Figure 3.19: The outlined orange rectangle with bold text shows the next case to be explored.

Proof. (*By Contradiction.*) Let I be the set of all fixed variables and let v_i be the value assigned to the variable x_i during search, for $i \in I$. Also let u_j^{CP} denote the updated upper bound that CPDEFAULT obtains and u_j^{BOX} be the updated upper bound that BOX obtains. For a contradiction, without loss of generality, assume that CP$_{\text{DEFAULT}}$ does more reduction on the upper bound of some variable $x_j, j \notin I$ than the BOX propagation and obtains u_j^{CP} where $u_j^{CP} < u_j^{BOX}$. Assuming that all variable bounds in this reduced rotated ellipsoid (when we fix $x_i = v_i, i \in I$) are redundant we know that $u_j^{CP} < u_j^B \leq u_j$ where u_j is the original upper bound of the variable x_j. Based on our assumption that the CP$_{\text{DEFAULT}}$ obtains a valid u_j^{CP} we know that the problem (3.4) is feasible. This implies that $v_i \in [l_i, u_i], \forall i \in I$.

Next, we consider the point x^B obtained as an optimal solution to the following problem:

$$\max_{x \in \mathbb{R}^n} \quad x_j^B \tag{3.12}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \left(\tilde{y}_i - \sum_{j \notin I} a_{ij} x_j \right)^2 \leq \beta \tag{3.13}$$

where $\tilde{y}_i := y_i - \sum_{j \in I} a_{ij} v_j$.

Notice that the obtained optimal point x^B lies in the full dimension ellipsoid and $x_j^B = u_j^{BOX}$. It follows that for any $k \notin I$, x_k^B satisfies the variable bounds as they are assumed to be redundant in the reduced ellipsoid (B.11) and x^B lies inside the ellipsoid. Moreover, as mentioned before $v_i \in [l_i, u_i], \forall i \in I$, thus x^B satisfies all variable bounds. Then x^B is also feasible for the CP$_{\text{DEFAULT}}$ problem (3.4). Our assumption was that CP$_{\text{DEFAULT}}$ propagation does strictly better than the BOX, which means it cut off this point. However CP$_{\text{DEFAULT}}$ propagation is a sound propagation algorithm, meaning it cannot remove any feasible solution such as this point thus we have a contradiction. □

Proposition 3.3.6. *There exist one-dimension smaller rotated hyper-ellipsoids with redundant bounds where BOX propagation* $\succ$ *CP*$_{\text{DEFAULT}}$.

Proof. The proof follows directly from Counterexample 3.3.1 from Proposition 3.3.1 as the bounds of all the variables are redundant and the value of 2 has been assigned to x_3. The details regarding the example are shown in Table 3.5, where the best bounds found by BOX propagation algorithm are highlighted.

Table 3.5: Bound comparison for $(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196$. The best bounds are highlighted.

	x_1		x_2		x_3	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
Original	-3	2	-2	3	2	2
Tangent box	-1.33	1.92	-1.67	2	2	2
$\text{CP}_{\text{DEFAULT}}$	-2	2	-2	2.50	2	2
BOX	-1.33	1.92	-1.67	2	2	2

□

Next we show that when we assign a value to one of the variables and we have at least one narrow bound in the reduced ellipsoid (Figure 3.20), then BOX and $\text{CP}_{\text{Default}}$ propagation algorithm are incomparable (Proposition 3.3.7).

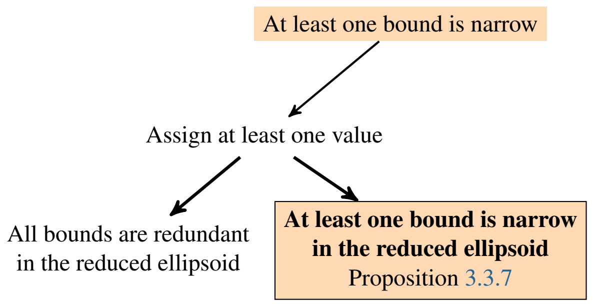

At least one bound is narrow
Assign at least one value
All bounds are redundant in the reduced ellipsoid
At least one bound is narrow in the reduced ellipsoid
Proposition 3.3.7

Figure 3.20: The outlined orange rectangle with bold text shows the next case to be explored.

Proposition 3.3.7. *The BOX propagation and the $\text{CP}_{\text{Default}}$ propagation are incomparable when we assign a value to one of the variables and at least one of the bounds in the resulting reduced rotated hyper-ellipsoid is narrow.*

Counterexample 3.3.5 (BOX $\succ$ $\text{CP}_{\text{DEFAULT}}$)**.** We use the same ellipsoid from Counterexample 3.3.1 in Proposition 3.3.1 but with a different set of starting variable bounds. Suppose we start with $x_1 \in [-1,2], x_2 \in [-4,4]$ and $x_3 \in [2,3]$, where x_1 has a narrow lower bound. Next we decide to assign the value 2 to x_3. The reduced ellipsoid has the form $(2-6x_1)^2+(3-(8x_1+4x_2))^2+(2-(2x_1+8x_2))^2 \leq 196$ where the lower bound of the variable x_1 is still narrow. The details regarding this example are shown in Table 3.6, where the bounds found by BOX are highlighted and the narrow bound is bold.

Counterexample 3.3.6 ($\text{CP}_{\text{DEFAULT}}$ $\succ$ BOX)**.** Next we present an example which shows that the $\text{CP}_{\text{DEFAULT}}$ propagation can do more inference than the BOX propagation. We consider the following ellipsoid constraint where x_2 has narrow bounds:

$$(2-x_1)^2+(5-(x_1+x_2+x_3))^2 \leq 8 \quad x_1 \in [-1,5], x_2 \in [3,5], x_3 \in [1,2]$$

Table 3.6: Bound comparison for $(2-6x_1)^2+(3-(8x_1+4x_2))^2+(2-(2x_1+8x_2))^2 \leq 196$. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
Original	**-1**	2	-4	4
Tangent box	**-1.33**	1.92	-1.67	2
CP$_{\text{DEFAULT}}$	-1	2	-2	2.25
BOX	-1	1.92	-1.67	2

Suppose now we assigned value 1 to x_3. The one-dimension-smaller ellipsoid has the form $(2-x_1)^2+(4-(x_1+x_2))^2 \leq 8$, where the bounds on x_2 are still narrow. In this reduced ellipsoid, the CP$_{\text{DEFAULT}}$ propagation does more inference on the upper bound of x_1. The details regarding this example are shown in Table 3.7, where the bounds found by CP$_{\text{DEFAULT}}$ are highlighted. The bounds calculated by BOX are calculated according to the problems (2.12) and (2.13). We then take the intersection of those bounds with the original bounds. The forward and backward propagation for CP$_{\text{DEFAULT}}$ are shown on Figure 3.21 and Figure 3.22, respectively.

Table 3.7: Bound Comparison for $(2-x_1)^2+(4-(x_1+x_2))^2 \leq 8$. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
Original	-1	5	3	5
Tangent box	-0.83	4.82	-2	6
CP$_{\text{DEFAULT}}$	-0.83	3.82	3	5
BOX	-0.83	4.82	3	5

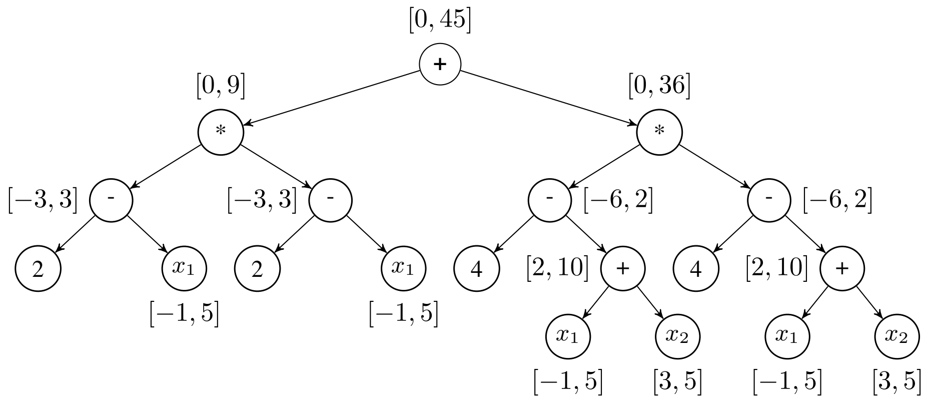


Figure 3.21: First Expression Tree (Forward) Propagation for $(2-x_1)^2+(4-(x_1+x_2))^2 \leq 8, x_1 \in [-1,5],\ x_2 \in [3,5]$.

Lastly we consider the case when we have at least one narrow bound in the rotated hyper-ellipsoid, for example a narrow upper bound, and decide not to assign any value. We know by definition (Definition 3.3.3) that we have a

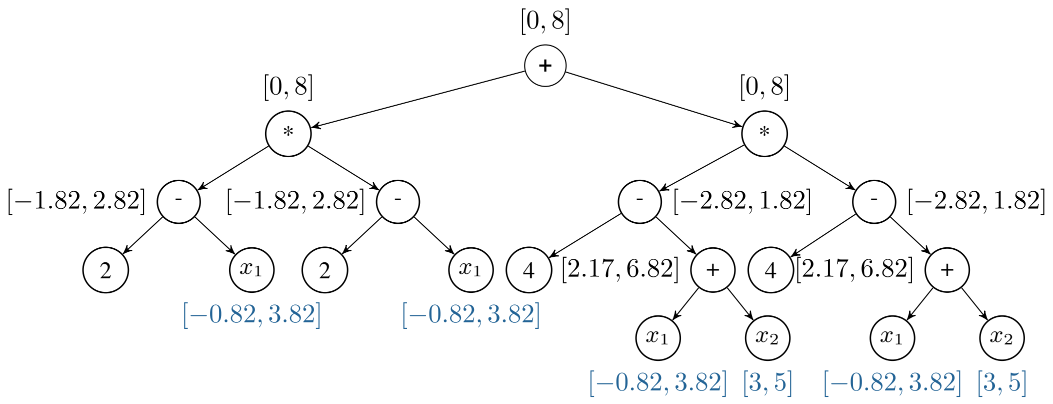


Figure 3.22: Final Expression Tree (Backward) Propagation for $(2 - x_1)^2 + (4 - (x_1 + x_2))^2 \leq 8, x_1 \in [-1, 5],\ x_2 \in [3, 5]$. The final reduced bounds are highlighted.

narrow upper bound if $u_j < u_j^B$. We also know that the BOX propagation finds u_j^{BOX} as $u_j^{BOX} = \min\{u_j, u_j^B\}$ which in this case equals to u_j. Since u_j is the best bound it can be obtained with the BOX propagation we see that the propagation algorithm does not do any pruning of the variable's upper bound and cannot do more propagation than the CP$_{\text{DEFAULT}}$ propagation algorithm to reduce the upper bound.

3.3.2 Integer Support

In this section we further discuss the findings of the analysis for when we have an integer support for variables for the different cases that we explored for rotated hyper-ellipsoid. We refer the reader to Appendix A and Appendix B for the expression trees and the detailed proofs, as we only present a general discussion in this section. Before going into greater details, recall the definition of integer support of variables (Definition (3.1.6)).

In Proposition 3.3.1, 3.3.4 and 3.3.7 we showed through a set of counterexamples that the BOX propagation and CP$_{\text{DEFAULT}}$ are incomparable for when we have rotated hyper-ellipsoid, when at least one bound is narrow in the rotated hyper-ellipsoid and when we have at least one narrow bound after we assign value to a variable, respectively. The details of the examples for when we have integer support are given in Table 3.8 and Table 3.9, where the best bounds are highlighted. The examples conclude the same as Proposition 3.3.1.

Table 3.8: Bound comparison for $(2 - 6x_1)^2 + (3 - (8x_1 + 4x_2))^2 + (14 - (2x_1 + 8x_2 + 6x_3))^2 \leq 196$ with *integer support* for variables. The best bounds are highlighted.

	x_1		x_2		x_3	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
Original	-3	2	-2	3	2	2
Tangent box	-1.33	1.92	-1.67	2	2	2
CP$_{\text{DEFAULT}}$	-2	2	-2	2	2	2
BOX	-1	1	-1	2	2	2

Table 3.9: Bound comparison for $(2 - x_1)^2 + (5 - (x_1 + x_2))^2 \leq 8$ with *integer support* for variables. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
Original	-1	5	5	7
Tangent box	-0.83	4.82	-1	7
CP_{DEFAULT}	0	2	5	7
BOX	0	4	5	7

Moreover, the two counterexamples with integer support presented in Table 3.9 and Table 3.10 make the same conclusions as Proposition 3.3.4 but for when we have integer support. The best bounds are highlighted. Likewise we can make the same conclusions as in Proposition 3.3.7, for when we have integer support for the variables, with the examples shown in Table 3.10 and Table 3.11.

Table 3.10: Bound comparison for $(2 - 6x_1)^2 + (3 - (8x_1 + 4x_2))^2 + (2 - (2x_1 + 8x_2))^2 \leq 196$ with *integer support* for variables. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
Original	-1	2	-4	4
Tangent box	-1.33	1.92	-1.67	2
CP_{DEFAULT}	-1	2	-2	2
BOX	-1	1	-1	2

Table 3.11: Bound comparison for $(2 - x_1)^2 + (4 - (x_1 + x_2))^2 \leq 8$ with *integer support* for variables. The best bounds are highlighted.

	x_1		x_2	
	Lower bound	Upper bound	Lower bound	Upper bound
Original	-1	5	3	5
Tangent box	-0.83	4.82	-2	6
CP_{DEFAULT}	0	3	3	5
BOX	0	4	3	5

Coming to the proof of Proposition 3.3.2, we never explicitly use anywhere the continuous support for the variables, but rather analyze the solutions of the optimization problems for $\text{CP}_{\text{DEFAULT}}$ and BOX propagation algorithms for rotated hyper-ellipsoids. In order to have integer support, we need to further restrict the optimization problems (3.4) and (3.8), meaning the domain of the variables to be the integer interval rather than continuous. The main argument that we are using to show that Proposition 3.3.2 is true, is that there exist a point x^B that cannot be removed by $\text{CP}_{\text{DEFAULT}}$, since the $\text{CP}_{\text{DEFAULT}}$ propagation algorithm is a sound algorithm. Accordingly we can show for the case of when we have integer support, that there exist a point x^B which is the optimal solution of the problem (3.8) where the domains of the variables are integer and all variables have integer support. The same proof approach and technique is used in Proposition 3.3.5 to show that the BOX propagation $\succeq$ $\text{CP}_{\text{DEFAULT}}$ once we assign a value to a variable and the bounds in the resulting one-dimension smaller ellipsoid are redundant. In case the point does not have integer support, then $\text{CP}_{\text{DEFAULT}}$ can successfully remove it thus doing more reduction than the BOX propagation algorithm as initially claimed.

In Table 3.8 we present the details for the example in Proposition 3.3.3, but with integer support for the variables where the best bounds are highlighted. We can see that we conclude the same result as in Proposition 3.3.3, which is that the BOX does more propagation than the $\text{CP}_{\text{DEFAULT}}$.

3.4 Discussion

The conducted analysis shows that there exist some cases where the BOX propagation is dominant over $\text{CP}_{\text{DEFAULT}}$ propagation. In other words, the BOX propagation is not completely dominant over the $\text{CP}_{\text{DEFAULT}}$ propagation when we have ellipsoid constrained problems and there exists a room for improvement. There has been constant work in the literature on speeding up [51, 56] as well as refining [12] constraint propagation algorithms. We believe our analysis is a good stepping stone to further develop more refined and efficient propagation algorithms for the ELLIPSOID global constraint.

Moreover, the analysis carried out in this chapter is relevant for understanding the strengths and weaknesses of different constraint propagation algorithms for the ELLIPSOID global constraint. By understanding the essence [4] of the BOX propagation algorithm for the ELLIPSOID global constraint, our formal analysis can serve as a *guide* for which problems the ELLIPSOID constraint is a suitable candidate if we know in advance on which types of problems the BOX propagation algorithm shows dominance.

Our analysis is consistent with most of the findings from Ku & Beck [32]. The BOX propagation algorithm performed better than $\text{CP}_{\text{DEFAULT}}$ propagation algorithm in terms of running time to prove optimality, on the binary quadratic programs, box-constrained integer least square programs and the ellipsoid-constrained integer least square programs. All of the problems include a rotated ellipsoid structure with redundant bounds for which, according to our theoretical analysis, the BOX propagation algorithm dominates over the $\text{CP}_{\text{DEFAULT}}$ propagation algorithm. On the quadratic lateness scheduling problems (QLSP) the $\text{CP}_{\text{DEFAULT}}$ propagation performed better than the BOX propagation algorithm. Due to the pure axis-aligned ellipsoid structure of the QLSP, according to our analysis the $\text{CP}_{\text{DEFAULT}}$ propagation algorithm dominates over the BOX propagation algorithm. The $\text{CP}_{\text{DEFAULT}}$ propagation algorithm also performed better than the BOX propagation algorithm on the exact quadratic knapsack problems, however the ellipsoid structure of these problems could not be determined from the problem data as it was done with the other problem sets.

The scope of our analysis focused on the cases in Figure 3.1. However, as a possible future research direction, it might be useful to look into the cases where the two propagation algorithms are incomparable and identify new subcases where a dominance can be proven.

3.5 Conclusion

In this chapter, we analyzed the strength of propagation algorithms for the ELLIPSOID constraint. Figure 3.23 shows a graphical summary of the results from the conducted analysis.

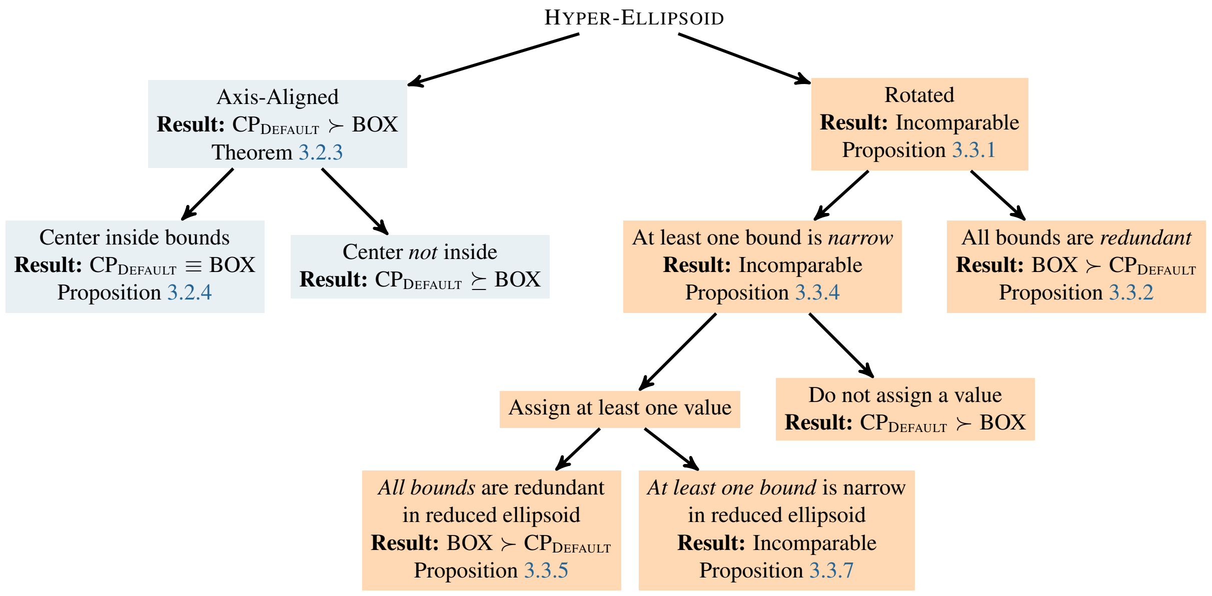

HYPER-ELLIPSOID
Axis-Aligned
Result: $\mathrm{CP}_{\textsc{Default}} \succ \mathrm{BOX}$
Theorem 3.2.3
Rotated
Result: Incomparable
Proposition 3.3.1
Center inside bounds
Result: $\mathrm{CP}_{\textsc{Default}} \equiv \mathrm{BOX}$
Proposition 3.2.4
Center *not* inside
Result: $\mathrm{CP}_{\textsc{Default}} \succeq \mathrm{BOX}$
At least one bound is *narrow*
Result: Incomparable
Proposition 3.3.4
All bounds are *redundant*
Result: $\mathrm{BOX} \succ \mathrm{CP}_{\textsc{Default}}$
Proposition 3.3.2
Assign at least one value
Do not assign a value
Result: $\mathrm{CP}_{\textsc{Default}} \succ \mathrm{BOX}$
All bounds are redundant in reduced ellipsoid
Result: $\mathrm{BOX} \succ \mathrm{CP}_{\textsc{Default}}$
Proposition 3.3.5
At least one bound is narrow in reduced ellipsoid
Result: Incomparable
Proposition 3.3.7

Figure 3.23: A Summary of the Formal Propagation Analysis for the ELLIPSOID constraint.

We discussed that the $\mathrm{CP}_{\textsc{Default}}$ propagation algorithm is dominant when we have a rotated hyper-ellipsoid where at least one of the variable bounds is narrow and no value has been assigned as well as for axis-aligned ellipsoids. We also discussed that if $\mathrm{CP}_{\textsc{Default}}$ is at least as good as the BOX propagation algorithm for axis-aligned ellipsoids, then the center does not belong to the domain hypercube. On the other hand, we proved that the BOX propagation is dominant when we have a rotated hyper-ellipsoid and all the bounds are redundant, and we keep assigning values, for when we have continuous and integer support.

In the next chapter, we apply the ELLIPSOID global constraint to a known problem from the literature and further investigate the performance and see if can gain more insights with empirical experiments.

Chapter 4

A Constraint Programming Approach for the Selective Tree Breeding Problem

In the previous chapter we analyzed the strength of constraint propagation for the ELLIPSOID constraint. In this chapter we explore the use of constraint programming as an approach for solving the selective tree breeding problem [44]: a problem studied in forest genetics and tree breeding literature in which a fixed-size tree breeding population needs to be determined from an available list of candidates such that the genetic value is maximized. We are particularly interested in this problem as there are no constraint programming approaches developed for solving it and the central constraint for maintaining genetic diversity is an ellipsoid constraint. We develop and compare three constraint programming (CP) approaches with a mixed-integer programming (MIP) formulation of the problem as a reference. The empirical analysis is conducted on instances extracted from the real-life data set of the Scots Pine case study in northern Sweden [45]. We show that our empirical evaluation is consistent with the theoretical results from Chapter 3. Specifically, the ELLIPSOID global constraint struggles on this problem likely due to the specific form of the constraint. We also discuss several possible approaches to improving the CP models.

Contributions. We develop an application of the ELLIPSOID global constraint, present a constraint programming (CP) approach for the selective tree breeding problem, and show that if further improved this approach could be competitive with existing MIP technology at least on small instances.

The chapter is organized as follows. In Section 4.1 we formally define the selective tree breeding problem and provide an overview of the relevant literature. In Section 4.2 we present the four approaches that we use to solve this problem. Section 4.3 describes our computational study of the performance of our proposed approaches. In Section 4.3.4 we present the empirical results, whereas in Section 4.4 we discuss the performance of the solution approaches. We end the chapter with some concluding remarks in Section 4.5.

4.1 Background

In this section, we present the formal definition of the selective tree breeding problem and review the techniques in the literature for solving it.

4.1.1 Problem Definition

A problem that arises in forest tree breeding is the optimal selection of candidates from a pedigree list in order to generate the best performance in seed orchards and at the same time maintain genetic diversity [28]. The trees which maximize the genetic value should be prioritized in the selection. However, they cannot be selected without considering the genetic relationship among them [37]. In general, tree breeders tend to avoid inbreeding by limiting the relatedness of individuals chosen for breeding.

Relatedness as a concept among populations can be defined in numerous ways, mainly using the notion of *identity by descent (IBD)*, a principal concept used in population genetics that determines genetically mediated similarities between individuals that are relatives [60]. Formally, for two individuals in a pedigree, IBD is defined as the event when both individuals inherited a proportion of their genes from the same ancestor [31]. Likewise, *coancestry* in a populations is defined as the proportion of genes of individuals which are identical by descent, meaning they are derived from the same ancestor [44]. To limit the relatedness of the breeding population, tree breeders want to control the coancestry under a certain threshold value.

A reduced genetic diversity among candidates might result in negative effects on the population such as reduced biodiversity as well as lower ability for the population to respond to changes in the environment [30]. For example, if the variation in genes is reduced, the population might have reduced ability to respond to new pests or future climate changes. Moreover the response will critically decrease as a result of accumulated coancestry. Organisms can also carry a certain number of lethal alleles[1] [69]. Strong inbreeding or self-fertilization among such individuals can lead to low adult fertility and a significant loss of offspring.

Figure 4.1 shows a simple example of the selective tree breeding problem. We have three individual trees, Tree #1, Tree #2 and Tree #3, that have different but beneficial genetic values. The first one, Tree #1, has a good vigour which is expressed in its *estimated breeding value* (EBV #1), while Tree #2 also has a good vigour in addition to the good form expressed as EBV #2. Lastly, Tree #3 has a great climate tolerance expressed as EBV #3. If we select Tree #2 and Tree #3 as candidates for breeding, then the resulting Tree #5 is likely to have good vigour and form and also great climate tolerance as its resulting EBV. Moreover it will maintain the genetic diversity unlike the resulting Tree #4 which most likely will only accumulate the genes for good vigour among the trees. Therefore we want to select the best candidates for breeding from a population that will give the highest genetic value while maintaining genetic diversity.

In (4.1), we provide the simplest optimization model for the selective tree breeding problem, where the con-

[1] An allele is one or more alternative forms of a gene that arise by a mutation.

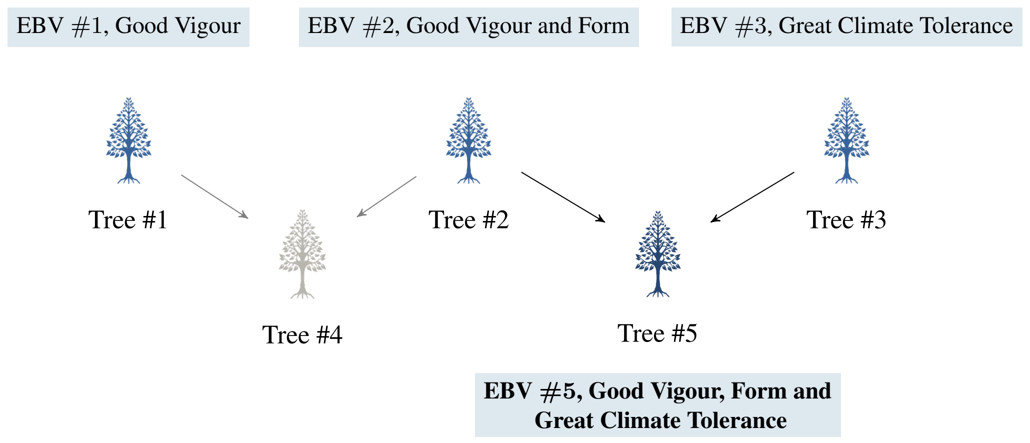

EBV #1, Good Vigour
EBV #2, Good Vigour and Form
EBV #3, Great Climate Tolerance
Tree #1
Tree #2
Tree #3
Tree #4
Tree #5
EBV #5, Good Vigour, Form and Great Climate Tolerance

Figure 4.1: A small example of the selective tree breeding problem, where we have two candidates selected for breeding (Tree #2 and Tree #3) to maximize the genetic value (EBV) and maintain genetic diversity.

tribution of all candidates is one, that was constructed by Meuwissen [41, 71].

$$\max_{\boldsymbol{x}} \quad \boldsymbol{g}^\top \boldsymbol{x} \tag{4.1a}$$

$$\text{subject to} \quad \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} \leq 2\theta \tag{4.1b}$$

$$\boldsymbol{1}^\top \boldsymbol{x} = 1 \tag{4.1c}$$

$$\boldsymbol{x}_j \in [lb_j, ub_j], \quad \forall j \in \{1, \dots, m\} \tag{4.1d}$$

In this model, m denotes the total number of candidates in the pedigree list. Note that $\boldsymbol{1} \in \mathbb{R}^m$ is the vector of ones, while $\boldsymbol{g} \in \mathbb{R}^m$ is a vector whose elements are the EBVs of all candidates. The variable vector $\boldsymbol{x}$ denotes the contribution of genes as a proportion from selected candidates. The objective function (4.1a) maximizes the EBVs of individuals in order to achieve the best genetic value. Constraint (4.1c) ensures that the overall contribution of candidates is one. In constraint (4.1d), $lb_j, ub_j \in \mathbb{R}$ correspond to the lower and upper bound of the variable x_j, respectively. In order to control the relatedness in the breeding population constraint (4.1b) is imposed on the *group coancestry*. Group coancestry in a population is the average coancestry between all individuals, including the coancestry of individuals with themselves [18]. The matrix $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ is called the numerator relationship matrix [70] of the pedigree. It represents the relatedness of individuals and its elements can be calculated efficiently with the formula introduced by Wright [70] as follows:

$$\begin{aligned} \boldsymbol{A}_{ij} = \boldsymbol{A}_{ji} &= \frac{\boldsymbol{A}_{j,p(i)} + \boldsymbol{A}_{j,q(i)}}{2} \quad && \forall i \in \{2, \dots, m\}, \ \forall j \in \{1, \dots, i-1\} \\ \boldsymbol{A}_{ii} &= 1 + \frac{\boldsymbol{A}_{p(i),q(i)}}{2} && \forall i \in \{1, \dots, m\} \end{aligned} \tag{4.2}$$

We use the notation $\boldsymbol{A}_{pq} = 0$ if $p = 0$ or $q = 0$.

We denote with $p(i)$ and $q(i)$ the two parents of individual i. Therefore, constraint (4.1b) restricts the group coancestry $\frac{x^\top Ax}{2}$ to be less than a given threshold value $\theta \in \mathbb{R}$.

Example. We present a simple example adapted from Yamashita et al. [71], where there is a pedigree list with five available candidates. We take a value of 0.625 as the threshold value for coancestry. We first show how the relationship matrix can be calculated. Figure 4.2 shows the relationship diagram between individuals while Table 4.1 presents the given data. We denote unknown parents with $p(i) = 0$ and $q(i) = 0$, respectively.

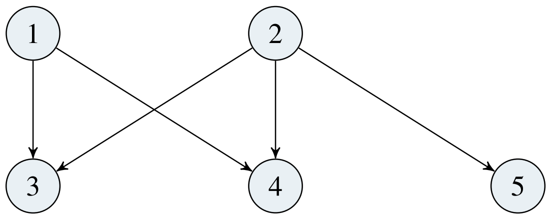


Figure 4.2: Relationship diagram for the data in Table 4.1, where the arrows point from parent to child.

Individual ID	Parent $p(i)$	Parent $q(i)$	EBV
1	0	0	80.4
2	0	0	0.0
3	1	2	90.4
4	1	2	-120.5
5	2	0	65.5

Table 4.1: An example of a given pedigree list.

By applying the formula from (4.2) to the example, we can compute the following values for $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0.5 & 0.5 & 0 \\ 0 & 1 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 & 0.25 \\ 0.5 & 0.5 & 0.5 & 1 & 0.25 \\ 0 & 0.5 & 0.25 & 0.25 & 1 \end{bmatrix}$$

The solution would be to pick the entire contribution from individual with ID = 3 for the breeding population:

$$x = (x_1, x_2, x_3, x_4, x_5) = (0, 0, 1, 0, 0)$$

It satisfies the principal constraint for genetic diversity (4.1b), the terms where the selected individual appears are underlined :

$$x_1 + x_2 + \underline{x_3} + x_4 + x_5 + \underline{0.5x_1x_3} + 0.5x_1x_4 + \underline{0.5x_2x_3} + 0.5x_2x_4 + 0.5x_2x_5 + \underline{0.5x_3x_1} + \underline{0.5x_3x_2} + \underline{0.5x_3x_4}$$
$$\underline{0.25x_3x_5} + 0.5x_4x_1 + 0.5x_4x_2 + \underline{0.5x_4x_3} + 0.25x_4x_5 + 0.5x_5x_2 + \underline{0.5x_5x_3} + 0.25x_5x_4 \leq 2 \times \mathbf{0.625}$$

This solution has the maximum EBV among the candidates: 90.4.

4.1.2 Literature Review

The use of optimization algorithms for solving the selective tree breeding problem has been increasing in the literature [2, 49]. Meuwissen [41], in addition to defining the model in (4.1) developed an optimal contribution algorithm (OC) used to solve the model efficiently. The algorithm itself is an iterative method based on Lagrange multipliers and has been widely used for multiple breeding applications, such as tree breeding management [27].

However, Pong-Wong and Woolliams [49] showed that the OC algorithm does not always attain optimal solution. Moreover, as an alternative, they formulated the problem as a semi-definite programming (SDP) problem. Since there are many efficient software packages developed for solving SDPs such as SeDuMi [57] and SDPT3 [61], Pong-Wong and Woolliams obtained the optimal solution.

Ahlinder et al. [2] employed a more robust SDP approach to optimize unequal contributions of genotypes. They defined additional operational constraints, as well as varying degrees of relatedness among candidates. They implemented this approach in a selection program called OPSEL [43] and solved large scale problems generated from real data. Even though their approach is guaranteed to find optimal solution, they reported a rather long computation time and required the candidate list to be shortened prior to computation.

Mullin and Belotti [44] tried to reduce computation time by developing a branch-and-bound algorithm. They combined their algorithm with an outer approximation method. Nevertheless, this method is still not very practical in computing solutions for large instances.

Recently, Yamashita et al. [71] developed a second-order cone programming approach for this problem. They reduced the computation time of the SDP approach successfully and attained optimal solutions.

Attempts to solve this problem with a constraint programming approach has not appeared in the literature. Our interest in the tree breeding problem was particularly sparked by the development of the ELLIPSOID global constraint [32], which is directly applicable to this problem.

4.2 Mathematical Models

In this section we present four approaches (one MIP and three CP models) to solve the selective tree breeding problem.

MIP Model. In (4.3) we present a MIP formulation which is consistent with the formulation of Belotti and Mullin (2016) [44].

$$\begin{aligned}
(\text{MIP})/(\text{CP}_{\text{DEFAULT}}): \quad & \max \quad \sum_{i=1}^{Z} \boldsymbol{g}_i \boldsymbol{x}_i && \text{(4.3a)}\\
& \text{subject to} \quad \sum_{i=1}^{Z}\sum_{j=1}^{Z} \boldsymbol{x}_i \boldsymbol{x}_j \boldsymbol{A}_{ij} \le \alpha && \text{(4.3b)}\\
& \sum_{i=1}^{Z} \boldsymbol{x}_i = N && \text{(4.3c)}\\
& \boldsymbol{x}_i \in \{0,1\}, \quad \forall i \in \{1,\dots,Z\} && \text{(4.3d)}
\end{aligned}$$

In this model, Z denotes the number of eligible candidates from the pedigee list, while N denotes the number of individuals to be selected for breeding. Objective (4.3a) maximizes the expected genetic advantages of contributions from the selected group of candidates. As described in Section 4.1.1, $\boldsymbol{g} \in \mathbb{R}^Z$ is a vector whose elements are the EBVs for all members. The contribution of genes as a proportion from the selected group is denoted by $\boldsymbol{c} \in \mathbb{R}^Z$. If selected, an individual contributes exactly $\frac{1}{N}$ of its genes, and if not selected contributes 0 thus $\boldsymbol{c}_i \in \{0, \frac{1}{N}\}$. In order to express the problem with binary variables, let $\boldsymbol{x} = N\boldsymbol{c}$. This way, an optimal solution of $\boldsymbol{x}$ gives also a solution for the vector $\boldsymbol{c}$. To control genetic diversity in the breeding population we impose constraint (4.3b) on the group coancestry θ, where $\alpha = 2\theta N^2$. Lastly, constraint (4.3c) ensures that we require a group of exactly N individual genotypes from the pedigree list with Z members that will contribute their genes in equal proportions.

CP Models. We use the formulation in (4.3) as our first CP formulation. If there is a MIP formulation of a problem, it is also a CP formulation of the same problem.

Since the relationship matrix, $\boldsymbol{A}$, in the relatedness constraint is always positive definite [49], we define a second formulation where we use the ELLIPSOID global constraint instead of the quadratic constraint on genetic diversity while we keep all of the other constraints:

$$(\text{CP}_{\text{ELLIPSOID}}): \quad \max \quad \sum_{i=1}^{Z} \boldsymbol{g}_i \boldsymbol{x}_i \tag{4.4a}$$

$$\text{subject to} \quad \text{ELLIPSOID}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i\}, \boldsymbol{A}, \alpha) \tag{4.4b}$$

$$\text{Constraints} \quad (4.3c) - (4.3d) \tag{4.4c}$$

In the ELLIPSOID constraint, the size of the ellipsoid is determined by α.

In our third CP model we add both the ELLIPSOID global constraint and the quadratic expression for the relatedness constraint:

$$(\text{CP}_{\text{BOTH}}): \quad \max \quad \sum_{i=1}^{Z} \boldsymbol{g}_i \boldsymbol{x}_i \tag{4.5a}$$

$$\text{subject to} \quad \text{Constraints} \quad (4.3b), (4.3c), (4.3d), (4.4b) \tag{4.5b}$$

4.3 Computational Study

In this section we present the results of our computational experiments to investigate the performance of the four solution approaches, namely MIP, CP$_{\text{DEFAULT}}$, CP$_{\text{ELLIPSOID}}$ and CP$_{\text{BOTH}}$. First we give our prediction of the results based on the analysis in Chapter 3. The next subsection presents the data set details and describes the problem instances. We then compare the performance of the proposed solution approaches.

4.3.1 Prediction of Performance

In our study in Chapter 3 (Section 3.3) we showed that the $\mathrm{CP_{DEFAULT}}$ and the BOX propagation algorithms are incomparable for rotated hyper-ellipsoids. Since the quadratic expression used in our models defines a rotated hyper-ellipsoid, we can expect that the $\mathrm{CP_{DEFAULT}}$ model and the $\mathrm{CP_{ELLIPSOID}}$ will be incomparable as well. Moreover, we know that the bounds on the variables are binary, meaning they are very narrow and lie inside the ellipsoid. Again, in our analysis in Chapter 3 (Section 3.3.1), we showed that when we have a rotated hyper-ellipsoid with narrow bounds, then the $\mathrm{CP_{DEFAULT}}$ and the BOX propagation algorithms are incomparable. Therefore we can expect the models $\mathrm{CP_{DEFAULT}}$ and $\mathrm{CP_{ELLIPSOID}}$ to be incomparable. Regarding our last model $\mathrm{CP_{Both}}$, where we use both propagations, we expect there will be an additional propagation effort which might lead to longer computation time.

4.3.2 Scots Pine Case Study Data Set

We demonstrate the application of the selective tree breeding problem by using real-data from a proposed Scots Pine breeding program in Northern Sweden [2, 45]. The data is available at the Dryad Digital Repository `https://doi.org/10.5061/dryad.4r1f0`.

In order to manage the accommodation to future climates from north to south of Sweden, a multiple-breeding population system called TREEPLAN has been developed with 24 closed breeding populations across Sweden [54]. The TREEPLAN system [40] gives ranked lists of candidates and their EBVs for a particular target region. Inside, there are over 90,000 records of genotypes from available field-test data, of which more than 43,000 candidates are alive and currently available as candidate genotypes.

Several selection tools and programs were developed to calculate the optimal genetic contribution for candidates so that the genetic gain is maximized while controlling inbreeding. One such example is GENCONT, a publicly available selection program using the OC algorithm [42]. However, many of these programs, including GENCONT [42], have memory limitations for the full list of 43,000 candidates when trying to solve the selective tree breeding problem [44]. Therefore in previous work [44], the candidate list was shortened to include only the best six individuals from each crossing (or interbreed). Finally, from the 43,000 candidates 5250 were chosen for the final list of candidates and are currently available at the Dryad Digital Repository as part of the final data set. We follow the same methodology in our computational study.

The data set is given as a *candidate file* for which an example is shown in Table 4.2. Each row in the table describes an individual with its ID, those of its Female and Male parents, its EBV, and finally the maximum number of times that the individual can occur in the selected population.

4.3.3 Experimental Setup

Instance Generation. From the provided Scots Pine case study data set we extracted 60 instances. The details regarding their sizes are given in Table 4.3 where the maximum size of a pedigree is 175 members. The instances

Table 4.2: A small example of a candidate file for the selective tree breeding problem. The Max column can contain either the value 1, showing that the individual is an eligible candidate, or 0, showing that the individual is not eligible, but is included as an ancestor to another individual (e.g., individual 2). The value 0 in the Female or Male column indicates that the particular parent is unknown.

Individual	**Female**	**Male**	**EBV**	**Max**
1	0	0	91.3	1
2	0	0	0.0	0
3	0	0	76.5	1
4	1	2	100.8	1
5	1	3	120.6	1

Table 4.3: Instances Sizes.

Number of instances	Z	N
10	50	10
10	75	20
10	100	50
10	120	50
10	150	50
10	175	50
Total: 60		

are generated in the following manner.

We first pick a random individual from the data set. Next, we check the set of selected individuals generated before, and add the new individual to the list if it has not been added. In order to find the next individual to pick, we check to see if the female parent of the selected individual has been added to the list. If the female parent is known and it has not been added to the list, then we take it as the next individual to explore. If it has been already added to the list, then we take the male parent, if known. In case both parents are unknown, then we take a new random individual from the data set. We stop when we reach the maximum generation[2] difference for ancestry we want to consider. In our experiments we consider a maximum generation difference of 10, meaning for any given individual in the set we take up to a maximum of 10 generations back.

We define $\mathcal{I}$ to be the entire (data) set of individual IDs available, $\mathcal{L}$ to be the set of generations, Z to be the minimum number of individuals to pick, $l(i) \in \mathcal{L}$ to be the generation of ancestors for individual $i \in \mathcal{I}$ and P to be the maximum generation difference for ancestry we want to consider. Moreover we define $f(i) \in \mathcal{I} \cup \{0\}$ as the *female* parent of individual $i \in \mathcal{I}$ while $m(i) \in \mathcal{I} \cup \{0\}$ denotes the *male* parent of individual $i \in \mathcal{I}$.

Algorithm 2 generates the list of Z individual IDs, which is $\mathcal{O}$, using the function FINDNEXTINDIVIDUAL() that is described further in Algorithm 3. Algorithm 2 starts by picking a random individual ID from $\mathcal{I}$ as shown in line 4. Then in line 6, the selected individual ID is added to $\mathcal{O}$ if it is not already there. The algorithm calls the function FINDNEXTINDIVIDUAL() in line 8 to get the next individual ID to explore. In line 9, if the

[2]A generation is a single stage or degree in the succession of natural descent. For example, father and son are *two* generations.

```
Algorithm 2: Generating a list of $Z$ individuals
1  Input: $\mathcal{I}, f, m, Z, P, l$
2  Let $\mathcal{O} = \{\}$
3  while ($|\mathcal{O}| < Z$) do
4      Pick a random integer $i$ from $\mathcal{I}$
5      if ($i \notin \mathcal{O}$) then
6          $\mathcal{O} = \mathcal{O} \cup \{i\}$
7          while true do
8              $j$ = FINDNEXTINDIVIDUAL ($i$)
9              if ($j = 0 \;\; || \;\; l(i) - l(j) > P$) then
10                 STOP
11             else
12                 $i = j$
13             end
14         end
15 end
16 Output: $\mathcal{O}$
```

Figure 4.3: Algorithm for generating a list of selected Z individuals.

```
Algorithm 3: FINDNEXTINDIVIDUAL()
1  Input: $i$
2  Let $j = 0$
3  if ($f(i) \neq 0$ & $f(i) \notin \mathcal{O}$) then
4      Let $j = f(i)$    → j is the next index to explore
5      $\mathcal{O} = \mathcal{O} \cup \{f(i)\}$
6      if ($m(i) \neq 0$ & $m(i) \notin \mathcal{O}$) then
7          $\mathcal{O} = \mathcal{O} \cup \{m(i)\}$
8  else if ($m(i) \neq 0$ & $m(i) \notin \mathcal{O}$) then
9      Let $j = m(i)$    → j is the next index to explore
10     $\mathcal{O} = \mathcal{O} \cup \{m(i)\}$
11 Output: $j$
```

Figure 4.4: Pseudocode for the function FINDNEXTINDIVIDUAL that takes an individual ID i and then returns the next individual ID j to explore.

next individual ID is 0 (unknown) or we are beyond the threshold value for maximum generation difference for ancestry then we stop, otherwise we assign the output individual ID from the function as the current individual ID. The algorithm then repeats the procedure until we reach at least Z individuals in $\mathcal{O}$, i.e., until the condition in the loop in line 3 is no longer true. Algorithm 3 accepts individual ID i and initializes the next individual to explore to its female parent in line 4 and it adds it to the list together with the male parent in lines 5 and 7, respectively, if they are not already there. If the female parent of individual i is already added to the list or it is unknown, then we initialize the next individual to explore to the male parent of individual i in line 9 and we add it to the list if it is not already there in line 10.

We estimate the threshold value for coancestry θ by heuristically generating T feasible solutions. Whenever we generate a new feasible solution, we calculate the corresponding threshold value θ for that solution according to the constraint for maintaining the group coancestry under a certain threshold value (4.3b), assuming we have obtained all the other values for the remaining variables before. Then we store the value in a vector. In the end we calculate θ as the average of all threshold values for the generated feasible solutions. We define T as the number of feasible solutions that we take, N is the number of individuals picked for the breeding population from Z while $\boldsymbol{y} \in \{0,1\}^Z$ is a random binary vector. This value vector is assigned to have exactly N ones and $(Z - N)$ zeros which are randomly ordered. Algorithm 4 provides the detailed steps.

First in lines 2 and 3, we calculate $\boldsymbol{A}$ assuming we already have the data from Algorithm 2, and then we set a null vector $\boldsymbol{R}$ for keeping the threshold values for each of the feasible solutions. In lines 4-7, we generate T feasible solutions i.e., binary vectors and calculate the threshold value for each. Lastly in line 8 we calculate θ as the average of all threshold values of the generated feasible solutions.

Algorithm 4: Generating $\boldsymbol{A}, \theta$

```
1 Input: $\mathcal{O}, T, N, Z$
2 Calculate $\boldsymbol{A}$ according to (4.2)
3 Let $\boldsymbol{R} = \{0\}^\top$
4 for $k = \{1, \ldots, T\}$ do
5     Generate a random binary vector $y \in \{0,1\}^Z$ such that $\mathbf{1}^\top$y=$N$
6     $\boldsymbol{R}[k] = \frac{y^\top \boldsymbol{A} y}{2N^2}$
7 end
8 $\theta = \frac{1}{T}\mathbf{1}^\top \boldsymbol{R}$
9 Output: $\boldsymbol{A}, \theta$
```

Figure 4.5: Algorithm for generating the threshold value θ for the group coancestry.

Software. We used the commercial solver CPLEX v12.6.3 to solve the MIP model and IBM ILOG CP Optimizer v12.6.3 to solve all the CP models. All experiments were run on a Red Hat Linux workstation with 8GB RAM and a 3.40 GHz Intel Core i7. We used a single thread and imposed a CPU time limit of 2000 seconds.

4.3.4 Experimental Results

Table 4.4 summarizes the results of all models on the generated instances. The MIP model solved all 60 instances with various sizes to optimality. Its fastest average running time is on the smallest set of instances with $Z = 50$ and $N = 10$. The CP$_{\text{DEFAULT}}$ model solves all 10 instances to optimality from the smallest instance set, and then struggles to compute solutions where the size of the pedigree list is greater than 120. Its average time increases as the size of the instances increases. The CP$_{\text{ELLIPSOID}}$ model solves only 7 instances to optimality for the smallest instance set. After $Z = 75$, it cannot prove optimality for any of the instances. The same is true for

Table 4.4: Number of instances solved to optimality and average runtime for the generated 60 instances from Table 4.3. The running times are reported in seconds. Bold numbers indicate the best approach for that instance set. The symbol '-' indicates that no problem instances were solved to optimality within 2000 seconds. Note that the average time is calculated over all the instances.

Instance Set		**MIP**		**CP$_{\text{DEFAULT}}$**		**CP$_{\text{ELLIPSOID}}$**		**CP$_{\text{BOTH}}$**	
Z	N	Average Runtime (sec.)	# Instances Solved to Optimality	Average Runtime (sec.)	# Instances Solved to Optimality	Average Runtime (sec.)	# Instances Solved to Optimality	Average Runtime (sec.)	# Instances Solved to Optimality
50	10	**0.009**	**10**	188.087	10	904.054	7	925.741	7
75	20	**0.026**	**10**	1815.226	1	-	-	-	-
100	50	**0.258**	**10**	1574.350	3	-	-	-	-
120	50	**0.053**	**10**	1801.344	1	-	-	-	-
150	50	**0.488**	**10**	-	-	-	-	-	-
175	50	**0.266**	**10**	-	-	-	-	-	-

the CP$_{\text{BOTH}}$ model. However, in this case we can observe that the average time is larger. This might be happening because we add the ELLIPSOID constraint as well as the quadratic expression in this model, as we are using the

BOX propagation algorithm on top of the CP$_{\text{DEFAULT}}$ propagation and this additional propagation effort adds more computation time.

We also calculate the mean relative error (MRE) of the three CP models across all instances using the formula $\left(\frac{z^* - z_{obs}}{z^*}\right) \times 100$. In this formula z_{obs} is the observed value for the objective, while z^* is the best objective value found by CPLEX. Figure 4.6 compares the MRE (%) of the three CP models over time. We observe that CP$_{\text{DEFAULT}}$ reduces the MRE below 10% within the first 0.15 seconds, while CP$_{\text{ELLIPSOID}}$ and CP$_{\text{BOTH}}$ reduce the MRE around 10% within the first 2.25 seconds.

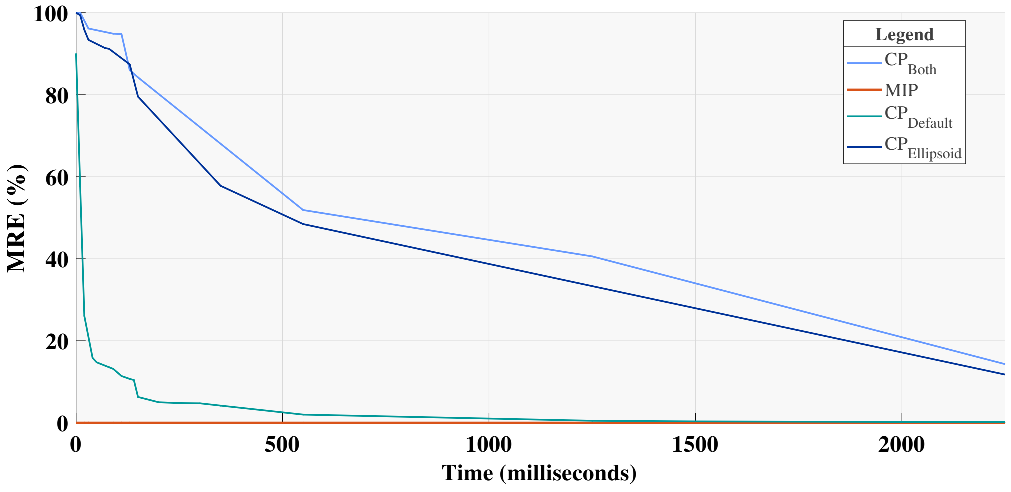


Figure 4.6: MRE (%) over time for the three CP models.

In Table 4.5 we show the results for the MRE over the instances for the three CP models. We observe that the average error shows that the objective is within 0.00013% of optimality for the first CP model throughout all the instances. For the second CP model it is within 0.51% and for the last (third) model it is within 0.0121% of optimality.

Table 4.5: Average MRE for the generated 60 instances from Table 4.3. Bold numbers indicate the smallest MRE for that instance set.

Instance Set		**CP$_{\text{DEFAULT}}$**	**CP$_{\text{ELLIPSOID}}$**	**CP$_{\text{BOTH}}$**
Z	N	Average MRE (%)	Average MRE (%)	Average MRE (%)
50	10	**0**	0.5808	**0**
75	20	**0**	0.2485	**0**
100	50	**0**	0.8137	**0**
120	50	**0.0008**	0.3062	**0.0008**
150	50	**0**	0.5661	**0**
175	50	**0**	0.6201	0.0682
Total Average:		**0.00013**	0.5100	0.0121

Variable and Value Ordering. We also experimented with variable and value ordering strategies on the middle sized instance set with $Z = 75$ and $N = 20$ to investigate the performance of the ELLIPSOID global constraint. We used all combinations of nondecreasing order, i.e., smallest to largest variable index/value and nonincreasing order, i.e., largest to smallest variable index/value. We compared all strategies to the discrete ellipsoid-based search (DEBS) [32], where we pick the value closest to the center of the ellipsoid. Although the optimality still could not be proven, we observed a different reduction in choice points. We see in Table 4.6 that the default ordering heuristic reduces the choice points the most, where among the other strategies, DEBS is the most competitive to the default ordering heuristic.

Table 4.6: Average number of choice points (chpts) for the proposed variable and value orderings, as well as the DEBS heuristic. The orderings were applied on the instance set with $Z = 75$ and $N = 20$, and on the $\text{CP}_{\text{ELLIPSOID}}$ model.

	Strategy 1	**Strategy 2**	**Strategy 3**	**Strategy 4**	**DEBS**	**DEFAULT**
Variable Ordering	Nondecreasing	Nondecreasing	Nonincreasing	Nonincreasing	-	-
Value Ordering	Nondecreasing	Nonincreasing	Nonincreasing	Nondecreasing	-	-
Average chpts:	2,314,929	2,796,324	2,469,452	2,850,865	1,925,145	**1,912,351**

4.4 Discussion

In this section, we discuss what we believe to be the main drawback for CP on the instances, particularly the large ones. Then we provide a discussion of future work for this research.

4.4.1 $\text{CP}_{\text{ELLIPSOID}}$ vs. $\text{CP}_{\text{DEFAULT}}$

In Section 4.3.1, we made a prediction about the performance of the models based on our study in Chapter 3 of the strength of the constraint propagation algorithms for the ELLIPSOID global constraint. We noticed that since the relationship matrix $\boldsymbol{A}$ describes a rotated ellipsoid, the $\text{CP}_{\text{DEFAULT}}$ and the BOX propagation algorithms are incomparable. However, we can have more insights from the variable bounds. As a subcase, we showed that when we have a rotated ellipsoid with narrow bounds, such as the binary variable bounds in this problem, if we assign a value to one of the variables, then the default CP propagation and the BOX propagation algorithms are still incomparable. It means that either the model with the ELLIPSOID constraint or the one without can make more inference. The experimental results in Table 4.4 show that for this particular problem the $\text{CP}_{\text{DEFAULT}}$ model without the ELLIPSOID constraint makes more inference and can find optimal solutions whereas the $\text{CP}_{\text{ELLIPSOID}}$ cannot, which is still consistent with our theoretical analysis. Overall, the results imply that for this particular problem, it is better to use the default propagation without adding the ELLIPSOID constraint.

4.4.2 The Impact of the Dual Bound

The empirical results indicate that the MIP technology is superior at proving optimality. One possible reason of the poor performance of CP, especially given the empirical results in Table 4.4 that shows that CP is able to find but not prove optimal solutions, is that it computes a weak dual bound. In order to gain more insights on this, we calculated the dual gap for the CP model according to the formula $\left(\frac{\boldsymbol{ub}(z)-\boldsymbol{lb}(z)}{z^*}\right) \times 100$. In the formula, z is the objective we are maximizing, while z^* is the optimal value found by CPLEX. The lower and upper bound of z are denoted with $\boldsymbol{lb}$, $\boldsymbol{ub}$ respectively. A bigger gap indicates the dual bound is very large, which also indicates a large search space of solutions for CP. We calculated and observed the dual gap after propagation at the root node. We report the average dual gap and the median for each of the instance sets in Figure 4.7.

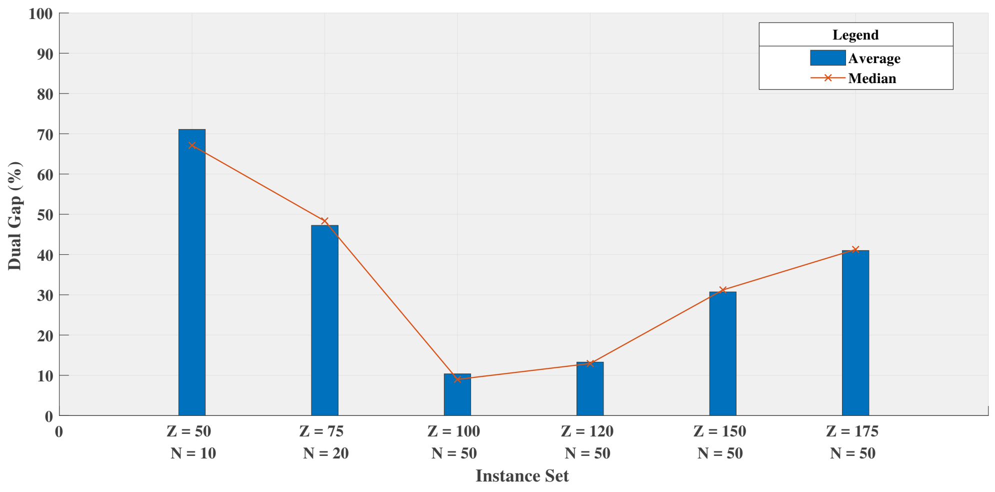


Figure 4.7: Dual gap (%) over the instances for CP.

One interesting observation that is evident from this figure is that the gap is very high for the instance sets with $Z = 50$ and $Z = 75$. Then it has lower values for when $Z = 100$ and after it goes slowly back up to higher values as we increase the size of the pedigree list. One possibility of why the gap is the lowest for the instances with $Z = 100$ is that for the particular instance set we have the smallest ratio of N to Z, which is $1 : 2$. The highest ratio on the other hand is for the first instance set where we have $1 : 5$ for $N = 10$ and $Z = 50$. To address our observations regarding the ratio, we generated additional instances for $Z = 50$ with $N = 20, 30, 40$, and for $Z = 100$ with $N = 20, 40, 60, 80$. The results are shown in Table 4.7.

Table 4.7: Average dual gap over the additional instances for $Z = 50$ with $N = 20, 30, 40$ and for $Z = 100$ with $N = 20, 40, 60, 80$. Bold numbers indicate the lowest value for the dual gap for that particular Z.

	Z = 50				**Z = 100**			
	N = 10	**N = 20**	**N = 30**	**N = 40**	**N = 20**	**N = 40**	**N = 60**	**N = 80**
Ratio: N to Z	1:5	2:5	3:5	4:5	1:5	2:5	3:5	4:5
Average Dual Gap (%)	71.09	16.66	**3.23**	4.28	80.89	16.03	**2.48**	7.23

We observe that for $Z = 50$ the dual gap declines to $N = 30$ and then increases. A similar pattern is seen for $Z = 100$. For both $Z = 50$ and $Z = 100$ the dual gap is the smallest for the ratio $3 : 5$ but not $4 : 5$. The particular pattern that we are seeing in the dual gap as we change the ratio is high-low-high. It resembles a hard-easy-hard pattern in search effort as the easy instances are associated with some sort of a phase transition and hard instances. Similar phase transitions of type easy-hard-easy [20] and easy-hard [5] have been described in the literature. In our case, we suspect that the constraindness on how many members from the candidate list will contribute their genes might account for the phase transition.

4.4.3 Future Work

Given the results from Section 4.3.4 we wish to develop a deeper understanding of why the CP approach is not performing well in order to improve its performance. Therefore, we identify two fronts that can be explored for future work.

- **Exploit Dominance**

  We could analyze the candidate trees and determine if one tree can be said to dominate another. As Belotti et al. [44] discussed, multiple individual trees can have the same parents. If that is the case then we would want to pick the tree that has a higher estimated breeding value before the other tree to attain larger objective function coefficients.

  Belotti et al. [44] define this constraint in the following manner. Suppose we have the entire set of candidates $\mathcal{I} = \{1, \dots, Z\}$ partitioned into K subsets such that $\mathcal{S}_1 \cup \mathcal{S}_2 \cdots \cup \mathcal{S}_K = \mathcal{I}$ and $\mathcal{S}_j \cap \mathcal{S}_k = \emptyset,$ for $\forall j \neq k$. Each subset is a set of trees with the same parents. Suppose also that all individuals in a subset are sorted in a nonincreasing order of their genetic values. Let $\mathcal{S}_k = \{i_1^k, \dots, i_{|\mathcal{S}_k|}^k\}$ for $k \in \{1, \dots, K\}$. Then, we can add the following constraints in order to focus only on the best solutions in the set:

$$x_{i_j^k} \geq x_{i_{j+1}^k} \quad \forall j \in \{1, \dots, |\mathcal{S}_k| - 1\}, \forall k \in \{1, \dots, K\} \tag{4.6}$$

  In this way it may be possible to reduce the search effort.

- **Create a Custom Cost Constraint**

  One common approach to improving a dual bound in a CP model is to create a custom global constraint which operates on the decision variables as well as the cost variable. Inside that constraint, the propagation algorithm calculates a bound on the cost variable and prunes its domain. For example, we can look at the *traveling salesman problem*. If we use a sequence constraint [67] we can prune the total cost of a route based on the solution to an assignment problem inside the constraint since that will give a lower bound on the tour length. As is already discussed in the literature [23, 24, 25] that same lower bound can be used to prune the domains and the cost variable, and also to do reduced cost fixing.

4.5 Conclusion

In this chapter, we investigated constraint programming approaches for solving the selective tree breeding problem [44] and compared their performance. Using test instances extracted from the Scots Pine Case Study dataset [45], we find that using the ELLIPSOID global constraint is not the best choice for this problem. Our computational results indicate that all of our constraint programming models perform worse than the MIP model. However we identify that this might be happening due to the fact that CP computes a weak dual bound. Therefore we also propose future directions for this research to improve the dual bound, such as exploiting dominance or creating a custom cost constraint.

In the next chapter, we summarize the contributions of this thesis and conclude.

Chapter 5

Conclusion

In this chapter we give the concluding remarks for the thesis. First we provide a summary of the work presented in the chapters and then conclude with possible future directions for research work.

5.1 Summary

An Analysis of Constraint Propagation Algorithms for the Ellipsoid Constraint. In Chapter 3 we performed a formal analysis of the strength of two constraint propagation algorithms, namely the BOX propagation and $\mathrm{CP}_{\textsc{Default}}$ propagation algorithm, for the ELLIPSOID global constraint. The main findings we obtained from the analysis are the following. The BOX propagation algorithm is dominant for rotated hyper-ellipsoid when all the variable bounds are redundant. Secondly, that the $\mathrm{CP}_{\textsc{Default}}$ propagation algorithm is at least as good as the BOX propagation algorithm for axis-aligned hyper-ellipsoids when the center of the ellipsoid is not inside the hypercube defined by the variable bounds and it is dominant for rotated hyper-ellipsoids where at least one of the variable bounds is narrow and no value has been assigned. The claims are valid both for continuous and integer support of variables.

A Constraint Programming Approach for the Selective Tree Breeding Problem. In Chapter 4 we used constraint programming approaches to solve the selective tree breeding problem. We explored the use of the ELLIPSOID global constraint for solving the problem. We extracted 60 instances from the Scots Pine Case Study dataset [45] to analyze the performance of the solution approaches. Our experimental findings indicated that the ELLIPSOID global constraint is not the best choice for this particular problem. The results showed that all of the constraint programming models performed worse than the MIP technology. With further analysis, we determined that this might be happening because of the weak dual bound that CP computes.

5.2 Future Work

Based on the research work from previous chapters, a number of directions can be explored for future work.

1. **Stronger Propagation Algorithms for the ELLIPSOID Global Constraint**

   The findings of the analysis demonstrate that there are cases where the BOX propagation dominates $\text{CP}_{\text{DEFAULT}}$ propagation algorithm but also there are some cases where $\text{CP}_{\text{DEFAULT}}$ is dominant over the BOX propagation algorithm, which indicates it is possible to further strengthen the BOX propagation algorithm. As mentioned in Section 4.4 there has been a steady work in the literature on speeding up [56] and also refining [12] constraint propagation algorithms. Our analysis offers key insights on how to further develop more refined and efficient propagation algorithms for the ELLIPSOID global constraint. For example, it shows that as the BOX propagation algorithm does not consider the variable bounds, it performs better when the problem is defined with redundant bounds. In fact, the main strength of the BOX propagation algorithm is that it calculates the reduced ellipsoid once a value is assigned to a variable, while $\text{CP}_{\text{DEFAULT}}$ propagation algorithm does not have information for the reduced ellipsoid. However, as $\text{CP}_{\text{DEFAULT}}$ propagation algorithm considers the variable bounds, it shows strength when the problem has narrow bounds. This implies that the information for the variable bounds is important in propagation algorithms for the ELLIPSOID constraint.

   Ku & Beck [32] investigated two other propagation algorithms for the ELLIPSOID global constraint: approximate bound consistency (ABC) and direct quadratically constrant programming (QCP). Even though the BOX propagation algorithm outperformed the other two filtering algorithms on the problem types investigated by Ku & Beck [32], it might be possible that ABC and QCP can perform better on other problem structures.

   Future research should also further investigate the cases when the propagation algorithms are incomparable, such as the case when we have a rotated ellipsoid with the initial bounds inside the tangent box that the BOX propagation calculates, and after we assign a value to one of the variables the resulting bounds in the reduced ellipsoid are again inside the tangent box. By exploring this particular case more, new subcases might arise where a dominance can be shown. For example it would be interesting to explore the dimension of the ellipsoid to see if any dominance can be proven in those subcases.

2. **Exploiting Dominance for the Selective Tree Breeding Problem**

   Considering the empirical results in Chapter 4 which showed that all CP models performed worse than the MIP models, we found that MIP outperforms CP because the latter computes a weak dual bound. To improve the performance of CP, one direction is to further analyze the dominance among candidate trees. In the case when multiple individual trees can have the same parents [44], the tree with a higher estimated breeding value should be prioritized before the other tree. Belotti et al. [44] discussed adding a constraint to the model in order to ensure the dominance is considered. It would be interesting to add this constraint to strengthen the CP model and possibly reduce the search effort. The details were given in Section 4.4.3.

3. **Custom Global Constraints and Multi-Valued Decision Diagrams**

   Another approach to improving the dual bound in a CP model is to create a custom global constraint. A cost-based global constraint operates on the decision variables as well as the cost variable. The propagation algorithm associated with the global constraint calculates a bound on the cost variable and therefore prunes its domain. One approach that is often used in custom global constraints as a dual bounding mechanism, like for example for the SEQUENCE() constraint, is multi-valued decision diagrams (MDDs) [8]. MDDs are an expressive data structure represent solutions of the constraints in a compact manner [3]. Taking MDDs into consideration as a technique that can refine the propagation and provide dual bounds in CP [9] is a promising direction for future research work

5.3 Concluding Remarks

The goal of this thesis is to formally analyze the strengths of two constraint propagation algorithms for the ELLIPSOID global constraint, namely the BOX propagation and $\text{CP}_{\text{DEFAULT}}$ propagation algorithm, so that more refined propagation algorithms for quadratic constraints can be designed. We presented an analysis of the strength of the propagation algorithms that points out there are still improvements to be made for the BOX propagation algorithm. We applied the ELLIPSOID global constraint to a known problem from the literature - the selective tree breeding problem - and saw that for this problem our empirical findings are consistent with the theoretical analysis. Specifically, as the ELLIPSOID constraint in the selective tree breeding problem describes a rotated ellipsoid with bounds inside the tangent BOX, our findings that $\text{CP}_{\text{DEFAULT}}$ performs better on the selective tree breeding problem are consistent with the conclusion from Proposition (3.3.7) (Section 3.3.1.1).

We believe that the research work presented in this thesis is a good stepping stone towards understanding propagation algorithms for quadratic constraints and designing stronger propagation algorithms for nonlinear constraints.

Appendices

Appendix A

Additional Expression Trees for Integer Support of Variables

A.1 Axis-Aligned Hyper-Ellipsoid

Below we present the expression trees for the example presented in Table 3.1.

A.2 Rotated Hyper-Ellipsoid

Below we present the expression trees for the examples:

1. Example presented in Table 3.8
2. Example presented in Table 3.9
3. Example presented in Table 3.10
4. Example presented in Table 3.11

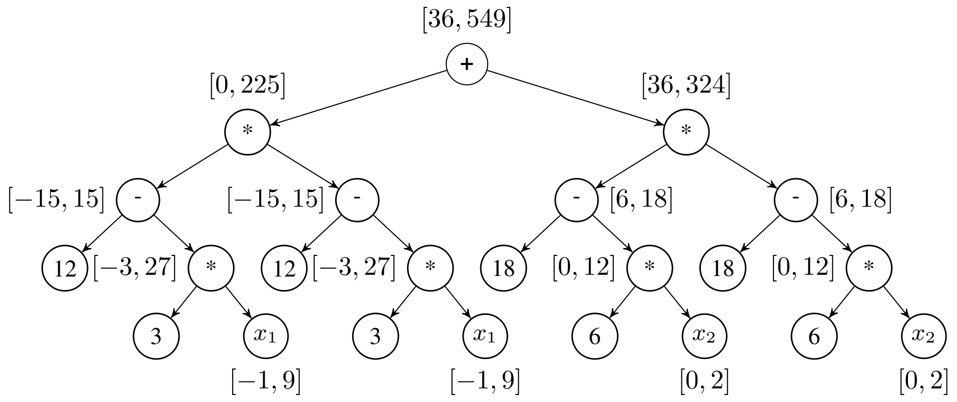


Figure A.1: First Expression Tree (Forward) Propagation for $(12-3x_1)^2+(18-6x_2)^2 \leq 144, x_1 \in [-1,9],\ x_2 \in [0,2]$.

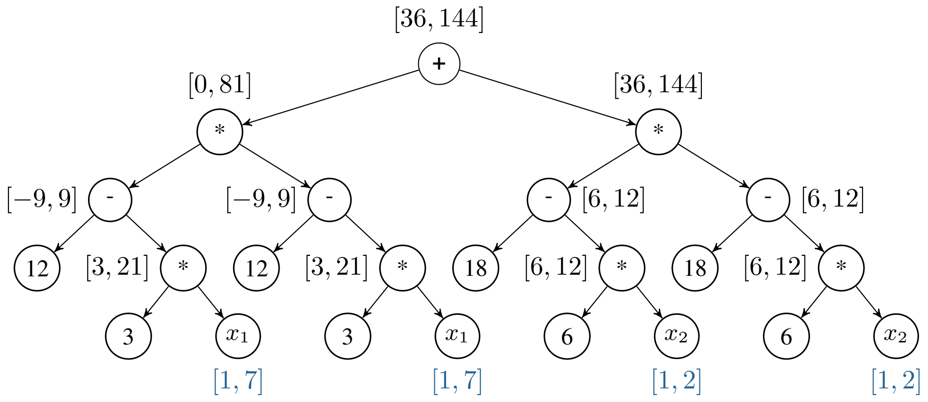


Figure A.2: Final Expression Tree (Backward) Propagation for $(12 - 3x_1)^2 + (18 - 6x_2)^2 \leq 144, x_1 \in [-1, 9],\ x_2 \in [0, 2]$.

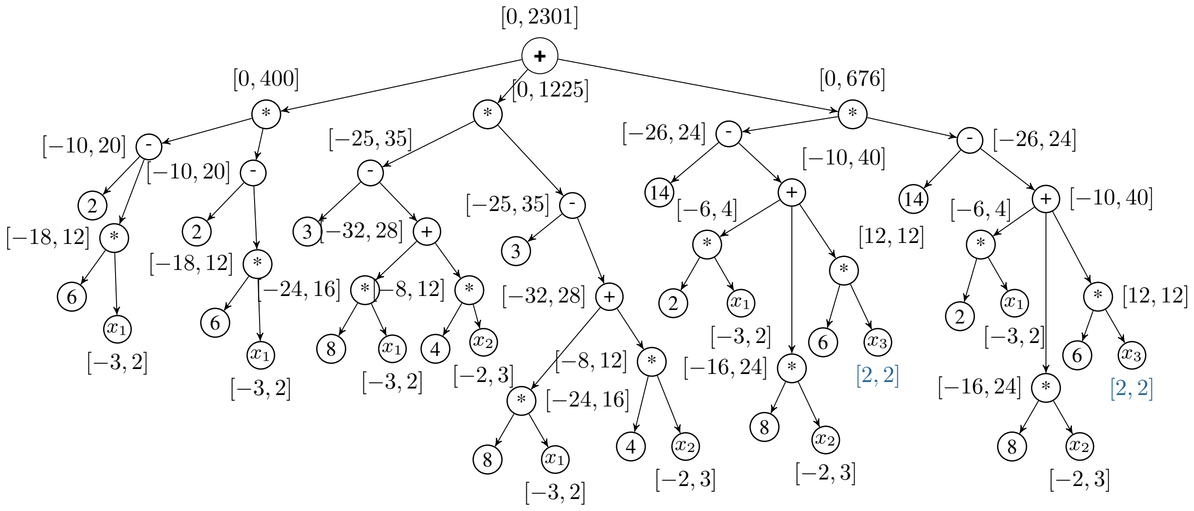


Figure A.3: First Expression Tree (Forward) Propagation for the ELLIPSOID:$(2-6x_1)^2 + (3-(8x_1+4x_2))^2 + (14-(2x_1+8x_2+6x_3))^2 \leq 196, x_1 \in [-3,2],\ x_2 \in [-2,3],\ x_3 \in [2,2]$.

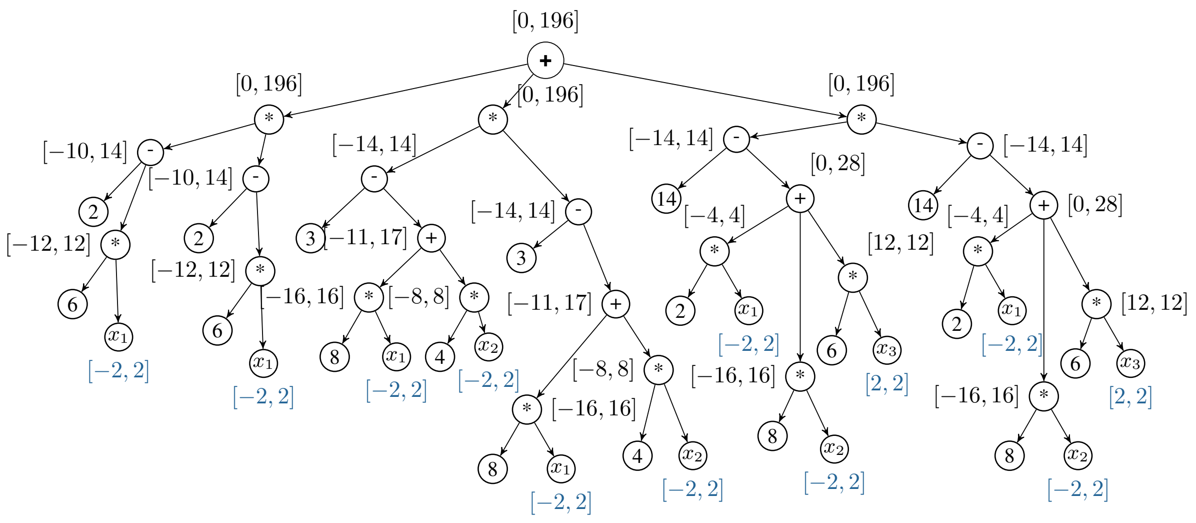


Figure A.4: Final Expression Tree (Backward) Propagation for the ELLIPSOID:$(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \leq 196, x_1 \in [-3,2],\ x_2 \in [-2,3],\ x_3 \in [2,2]$.

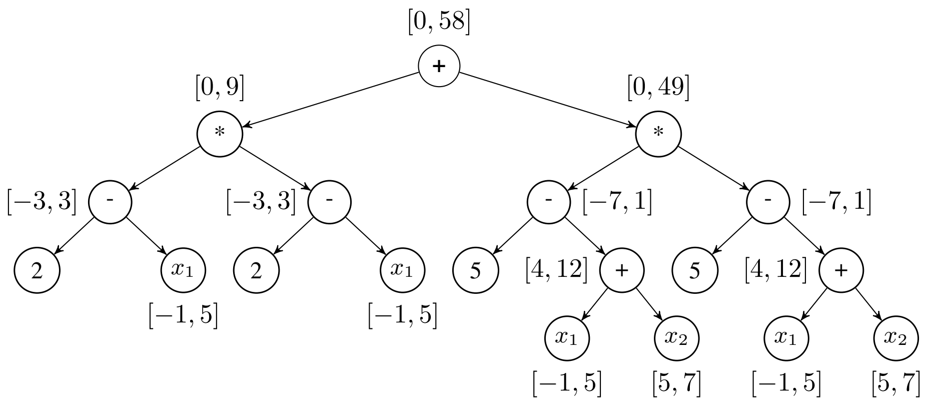


Figure A.5: First Expression Tree (Forward) Propagation for $(2-x_1)^2+(5-(x_1+x_2))^2 \leq 8, x_1 \in [-1,5],\ x_2 \in [5,7]$.

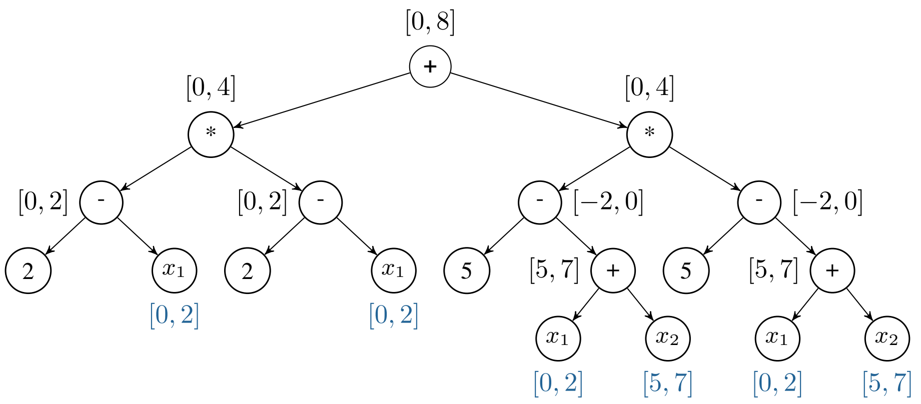


Figure A.6: Final Expression Tree (Backward) Propagation for $(2-x_1)^2+(5-(x_1+x_2))^2 \leq 8, x_1 \in [-1,5],\ x_2 \in [5,7]$.

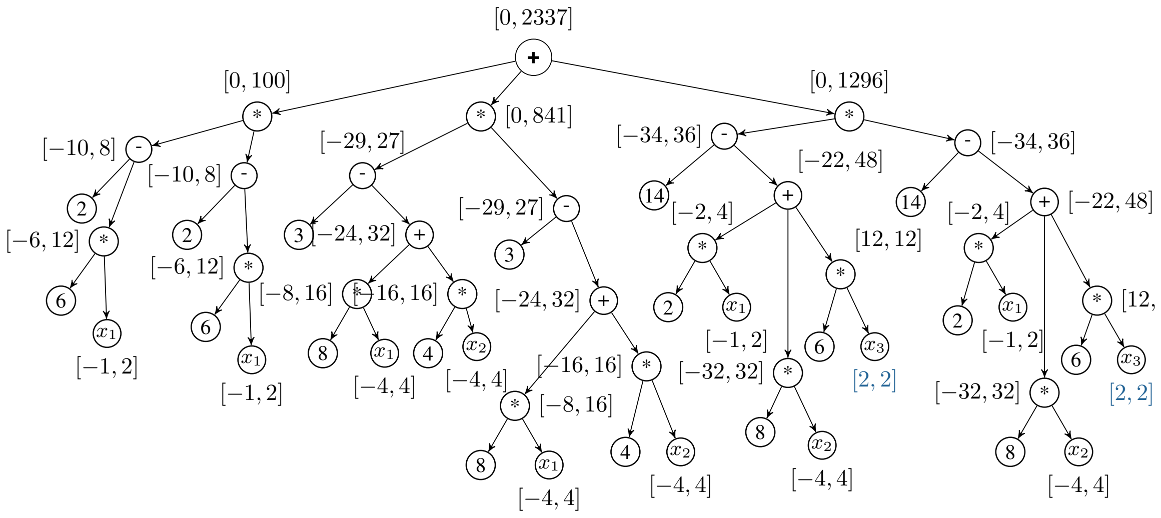


Figure A.7: First Expression Tree (Forward) Propagation for the ELLIPSOID: $(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \le 196, x_1 \in [-1,2],\ x_2 \in [-4,4],\ x_3 \in [2,2]$.

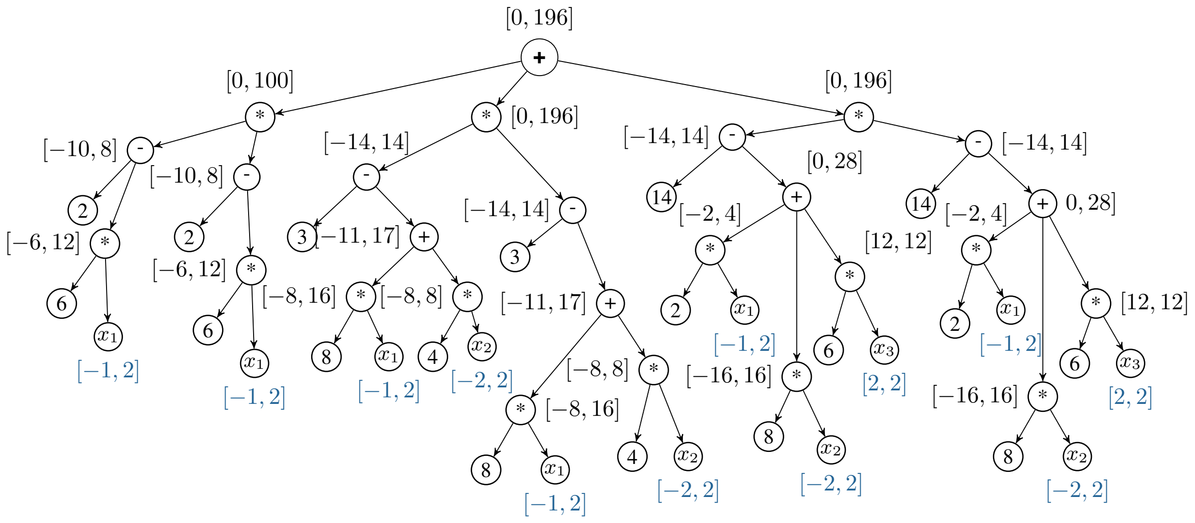


Figure A.8: Final Expression Tree (Backward) Propagation for the ELLIPSOID: $(2-6x_1)^2+(3-(8x_1+4x_2))^2+(14-(2x_1+8x_2+6x_3))^2 \le 196, x_1 \in [-1,2],\ x_2 \in [-4,4],\ x_3 \in [2,2]$.

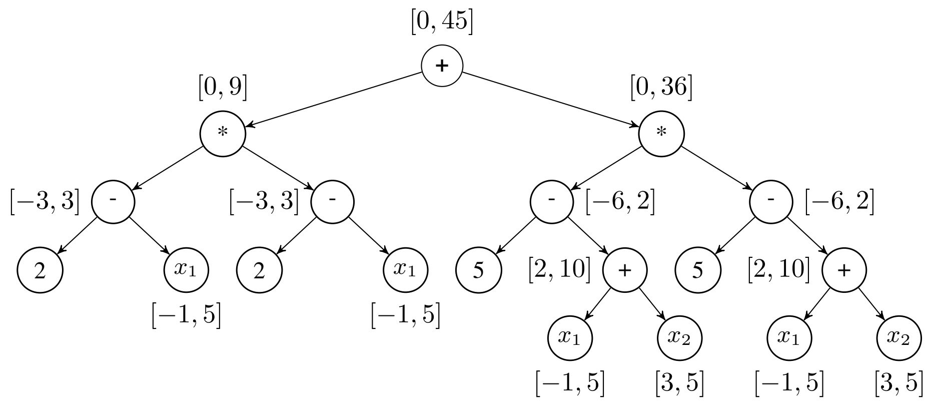


Figure A.9: First Expression Tree (Forward) Propagation for $(2-x_1)^2+(4-(x_1+x_2))^2 \leq 8, x_1 \in [-1,5],\ x_2 \in [3,5]$.

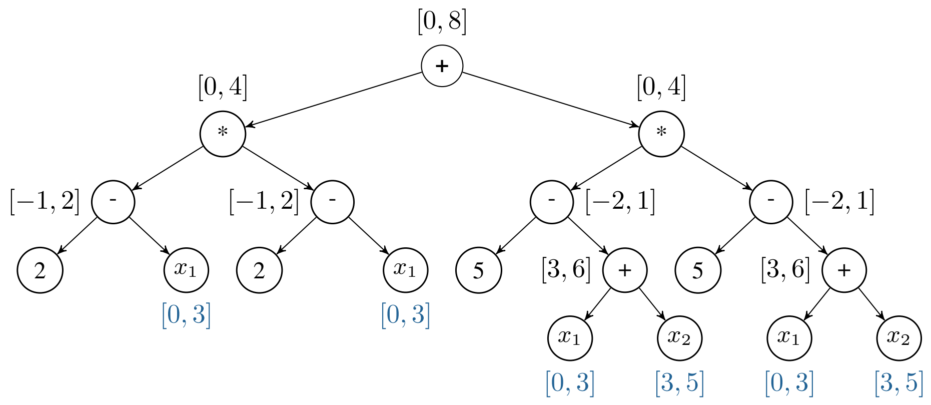


Figure A.10: Final Expression Tree (Backward) Propagation for $(2-x_1)^2+(4-(x_1+x_2))^2 \leq 8, x_1 \in [-1,5],\ x_2 \in [3,5]$.

Appendix B

Additional Proofs for Integer Support of Variables

B.1 Axis-Alligned Hyper-Ellipsoid

$$\begin{aligned} l_j^{CP} = \min\ & x_j \\ \text{s.t.}\ & (y_j - a_{jj}x_j)^2 \leq \beta - \sum_{i \in \{1,\dots n\} \setminus \{j\}} \min_{l_i \leq x_i \leq u_i} (y_i - a_{ii}x_i)^2 \\ & l_j \leq x_j \leq u_j \end{aligned} \tag{B.1}$$

$$u_j^{CP} = \max\ x_j \tag{B.2a}$$

$$\text{s.t.}\ (y_j - a_{jj}x_j)^2 \leq \beta - \sum_{i \in \{1,\dots n\} \setminus \{j\}} \min_{l_i \leq x_i \leq u_i} (y_i - a_{ii}x_i)^2 \tag{B.2b}$$

$$l_j \leq x_j \leq u_j \tag{B.2c}$$

$$\begin{aligned} l_j^B = \min_{x \in \mathbb{Z}^n}\ & x_j \\ \text{s.t.}\ & \sum_{i=1}^{n} (y_i - a_{ii}x_i)^2 \leq \beta \end{aligned} \tag{B.3}$$

$$\begin{aligned} u_j^B = \max_{x \in \mathbb{Z}^n}\ & x_j \\ \text{s.t.}\ & \sum_{i=1}^{n} (y_i - a_{ii}x_i)^2 \leq \beta \end{aligned} \tag{B.4}$$

Lemma B.1.1. *(Corresponding to Lemma 3.2.1) For axis-aligned hyper-ellipsoids with integer support of variables, CP*$_{\text{DEFAULT}} \succeq$ *BOX propagation.*

Proof. Let x_j be an arbitrary variable with integer domain $[l_j, u_j]$. Consider the domains $[l_j^{CP}, u_j^{CP}]$ and $[l_j^{BOX}, u_j^{BOX}]$ obtained after applying CP$_{\text{DEFAULT}}$ and BOX propagation algorithms, respectively. We first show that $u_j^{CP} \leq u_j^{BOX}$.

Consider the optimization problems defined with (B.2) and (B.4). We will show that (B.4) is a relaxation of (B.2). Let $\tilde{x}_j$ be a feasible solution to (B.2). Also, let $\tilde{x}_i$ be an optimal solution to the minimization problem in the right-hand-side of (B.2b). Then, $\tilde{x}$ is feasible to (B.4) as we have:

$$(y_j - a_{jj}\tilde{x}_j)^2 \leq \beta - \sum_{i\in\{1,\dots n\}\setminus\{j\}} (y_i - a_{ii}\tilde{x}_i)^2 \Rightarrow \sum_{i=1}^{n}(y_i - a_{ii}\tilde{x}_i)^2 \leq \beta \tag{B.5}$$

Note that the objective functions of (B.2) and (B.4) are the same.

Therefore, we obtain $u_j^{CP} \leq u_j^B$. We also know that $u_j^{CP} \leq u_j$ due to (3.2c), so we get $u_j^{CP} \leq \min\{u_j, u_j^B\} = u_j^{BOX}$.

Using similar arguments, it is easy to show that (3.5) is a relaxation of (3.1), thus $l_j^{CP} \geq l_j^{BOX}$, which concludes the proof. □

Lemma B.1.2. *(Corresponding to Lemma 3.2.2) There exists axis-aligned hyper-ellipsoids with integer support of variables where CP*$_{\text{DEFAULT}}$ $\succ$ *BOX propagation.*

Proof. Example is given in Table 3.1. □

Theorem B.1.3. *(Corresponding to Theorem 3.2.3) For axis-aligned hyper-ellipsoids with integer support of variables CP*$_{\text{DEFAULT}}$ $\succ$ *BOX propagation.*

Proof. Follows directly from Lemma B.1.1 and Lemma B.1.2. □

Proposition B.1.4. *(Corresponding to Proposition 3.2.4) If the integer domain hypercube includes the centre of the ellipsoid then CP*$_{\text{DEFAULT}}$ $\equiv$ *BOX propagation algorithm for axis-aligned hyper-ellipsoids.*

Proof. Let x_j be an arbitrary variable with integer domain $[l_j, u_j]$. Consider the domains $[l_j^{CP}, u_j^{CP}]$ and $[l_j^{BOX}, u_j^{BOX}]$ obtained after applying CP$_{\text{DEFAULT}}$ and BOX propagation algorithms, respectively. Also suppose that the center of the ellipsoid belongs to the integer domain hypercube. We first show that $u_j^{CP} = u_j^{BOX}$.

Consider the optimization problems defined with (B.2) and (B.4). Since we aim at maximizing x_j we need to maximize the right-hand-side of both (3.2) and (3.6). As (3.6) is defined over $\mathbb{Z}^n$, the minimum value that we can obtain for the sum of the quadratic expressions subtracted from β, attained at $x_i = \frac{y_i}{a_{ii}}, \forall i \in \{1, \dots, n\}\setminus\{j\}$, is 0. So for (3.6) we get the following form:

$$u_j^B = \max_{x_j\in\mathbb{R}} x_j \quad \text{s.t.} \quad (y_j - a_{jj}x_j)^2 \leq \beta$$

Thus for the optimal solution of (3.6) we get:

$$u_j^B = \frac{y_j}{a_{jj}} + \left|\frac{\sqrt{\beta}}{a_{jj}}\right|$$

Next, let $\delta_j := \sum_{i\in\{1,\dots,n\}\setminus\{j\}} \min_{l_i\leq x_i\leq u_i}(y_i - a_{ii}x_i)^2$. According to our initial assumption that the center of the ellipsoid belongs to the domain hypercube, i.e., $\frac{y_i}{a_{ii}} \in [l_i, u_i]$ it follows that $\delta_j = 0$. That is, (B.2) is equivalent to :

$$u_j^{CP} = \max_{l_j \le x_j \le u_j} \; x_j$$
$$\text{s.t. } (y_j - a_{jj}x_j)^2 \le \beta$$

Then for the optimal solution we get:

$$\begin{aligned} u_j^{CP} &= \min\left\{u_j, \frac{y_j}{a_{jj}} + \left|\frac{\sqrt{\beta}}{a_{jj}}\right|\right\} \\ &= \min\{u_j, u_j^B\} = u_j^{BOX} \end{aligned}$$

Using similar arguments, it is easy to show that $l_j^{CP} = l_j^{BOX}$, which concludes the proof. □

B.2 Rotated Hyper-Ellipsoid

$$\begin{aligned} l_k^{CP} = \min_{x\in\mathbb{Z}^n} \quad & x_k \\ \text{s.t.} \quad & \left(y_s - \sum_{j\in N} a_{sj}x_j\right)^2 \le \beta - \sum_{i\in N\setminus S^k} \min_{\substack{l_j \le x_j \le u_j \\ j\in N}} \left(y_i - \sum_{j\in J} a_{ij}x_j\right)^2, \forall s \in S^k \qquad \text{(B.6)} \\ & l_i \le x_i \le u_i, \forall i \in N \end{aligned}$$

$$u_k^{CP} = \max_{x\in\mathbb{Z}^n} \quad x_k \tag{B.7a}$$
$$\text{s.t.} \quad \left(y_s - \sum_{j\in N} a_{sj}x_j\right)^2 \le \beta - \sum_{i\in N\setminus S^k} \min_{\substack{l_j \le x_j \le u_j \\ j\in N}} \left(y_i - \sum_{j\in J} a_{ij}x_j\right)^2, \forall s \in S^k \tag{B.7b}$$
$$l_i \le x_i \le u_i, \forall i \in N \tag{B.7c}$$

$$\begin{aligned} l_p^B = \min_{x\in\mathbb{Z}^n} \; & x_p \\ \text{s.t. } & \sum_{i=1}^n (y_i - \sum_{j=1}^n a_{ij}x_j)^2 \le \beta \end{aligned} \qquad \text{(B.8)} \qquad\qquad \begin{aligned} u_p^B = \max_{x\in\mathbb{Z}^n} \; & x_p \\ \text{s.t. } & \sum_{i=1}^n (y_i - \sum_{j=1}^n a_{ij}x_j)^2 \le \beta \end{aligned} \qquad \text{(B.9)}$$

Proposition B.2.1. *(Corresponding to Proposition 3.3.2) If all variable bounds in the rotated hyper ellipsoid are redundant and all variables have integer support then BOX propagation $\succeq$ CP*$_{\text{DEFAULT}}$.

Proof. (By Contradiction.) Let u_j^{CP} denote the updated upper bound that CP$_{\text{DEFAULT}}$ obtains and u_j^{BOX} be the updated upper bound that BOX obtains. For a contradiction, without loss of generality, suppose that CP$_{\text{DEFAULT}}$ does more reduction on the upper bound of some variable $x_j, j \in \{1, \dots n\}$ than the BOX propagation and obtains

u_j^{CP} where $u_j^{CP} < u_j^{BOX}$. Assuming that all variable bounds are redundant we know that $u_j^{CP} < u_j^B \leq u_j$ where u_j is the original upper bound of the variable x_j. Based on our assumption that the CP$_{\text{DEFAULT}}$ obtains a valid u_j^{CP} we know the problem (3.4) is feasible. This also implies that the problem (3.8) is feasible. Next, we consider the point x^B obtained as an optimal solution to the feasible BOX problem (3.8), where $x_j^B = u_j^B$. The point x^B satisfies all variable bounds as they are all redundant and x^B lies inside the ellipsoid. This point is also feasible to the CP$_{\text{DEFAULT}}$ problem (3.4) as it lies inside the ellipsoid, meaning it satisfies the ELLIPSOID constraint. Our initial assumption was that CP$_{\text{DEFAULT}}$ propagation does strictly better than the BOX, which means it cuts off this point. However the CP$_{\text{DEFAULT}}$ propagation is a sound propagation algorithm, meaning it cannot remove any feasible solution such as this point thus we have a contradiction. □

Proposition B.2.2. *(Corresponding to Proposition 3.3.3) There exist rotated hyper-ellipsoids with redundant bounds and integer support of variables where BOX propagation $\succ$ CP$_{\text{DEFAULT}}$.*

Proof. Example is given in Table 3.8. □

Proposition B.2.3. *(Corresponding to Proposition 3.3.5) If there is at least one fixed variable and all the bounds in the resulting reduced rotated hyper-ellipsoid are redundant and all have integer support, then BOX propagation $\succeq$ CP$_{\text{DEFAULT}}$.*

Proof. (*By Contradiction.*) Let I be the set of all fixed variables and let v_i be the value assigned to the variable x_i during search, for $i \in I$. Also let u_j^{CP} denote the updated upper bound that CP$_{\text{DEFAULT}}$ obtains and u_j^{BOX} be the updated upper bound that BOX obtains. For a contradiction, without loss of generality, assume that CP$_{\text{DEFAULT}}$ does more reduction on the upper bound of some variable $x_j, j \notin I$ than the BOX propagation and obtains u_j^{CP} where $u_j^{CP} < u_j^{BOX}$. Assuming that all variable bounds in this reduced rotated ellipsoid (when we fix $x_i = v_i, i \in I$) are redundant we know that $u_j^{CP} < u_j^B \leq u_j$ where u_j is the original upper bound of the variable x_j. Based on our assumption that the CP$_{\text{DEFAULT}}$ obtains a valid u_j^{CP} we know that the problem (3.4) is feasible. This implies that $v_i \in [l_i, u_i], \forall i \in I$.

Next, we consider the point x^B obtained as an optimal solution to the following problem:

$$\max_{x \in \mathbb{Z}^n} \quad x_j^B \tag{B.10}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \left(\tilde{y}_i - \sum_{j \notin I} a_{ij} x_j \right)^2 \leq \beta \tag{B.11}$$

where $\tilde{y}_i := y_i - \sum_{j \in I} a_{ij} v_j$.

Notice that the obtained optimal point x^B lies in the full dimension ellipsoid and $x_j^B = u_j^{BOX}$. It follows that for any $k \notin I$, x_k^B satisfies the variable bounds as they are assumed to be redundant in the reduced ellipsoid (B.11) and x^B lies inside the ellipsoid. Moreover, as mentioned before $v_i \in [l_i, u_i], \forall i \in I$, thus x^B satisfies all variable bounds. Then x^B is also feasible for the CP$_{\text{DEFAULT}}$ problem (3.4). Our assumption was that CP$_{\text{DEFAULT}}$ propagation does strictly better than the BOX, which means it cut off this point. However CP$_{\text{DEFAULT}}$ propagation is a sound propagation algorithm, meaning it cannot remove any feasible solution such as this point thus we have a contradiction. □

Bibliography

[1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201 – 2214, 2002.

[2] J. Ahlinder, T.J. Mullin, and M. Yamashita. Using semidefinite programming to optimize unequal deployment of genotypes to a clonal seed orchard. *Tree Genetics & Genomes*, 10(1):27–34, 2014.

[3] 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.

[4] K. R. Apt. The Essence of Constraint Propagation. *Theoretical computer science*, 221(1-2):179–210, 1999.

[5] C. Bäckström and P. Jonsson. Time and space bounds for planning. *Journal of Artificial Intelligence Research*, 60:595–638, 2017.

[6] J. C. Beck, A. J. Davenport, E. D. Davis, and M. Fox. The ODO project: Toward a unified basis for constraint-directed scheduling. *Journal of Scheduling*, 1(2):89–125, 1998.

[7] N. Beldiceanu, M. Carlsson, S. Demassey, and T. Petit. Global Constraint Catalogue: Past, Present and Future. *Constraints*, 12(1):21–62, 2007.

[8] D. Bergman, A. A. Ciré, and W.-J. van Hoeve. MDD Propagation for Sequence Constraints. *Journal of Artificial Intelligence Research*, 50:697–722, 2014.

[9] D. Bergman, A. A. Ciré, W.-J. van Hoeve, and J. N. Hooker. *Decision Diagrams for Optimization*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, 2016.

[10] C. Bessière. *Handbook of constraint programming – Chapter 3, Constraint Propagation*. Elsevier Science, 2006.

[11] C. Bessière and J.-C. Régin. Enforcing Arc Consistency on Global Constraints by Solving Subproblems on the Fly. *International Conference on Principles and Practice of Constraint Programming*, pages 103–117, 1999.

[12] C. Bessière and J.-C. Régin. Refining the Basic Constraint Propagation Algorithm. In *International Joint Conference on Artificial Intelligence, IJCAI*, volume 1, pages 309–315, 2001.

[13] V. Bier. Game-Theoretic and Reliability Methods in Counterterrorism and Security. In *Statistical Methods in Counterterrorism: Game Theory, Modeling, Syndromic Surveillance, and Biometric Authentication*, pages 23–40. Springer, 2006.

[14] 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.

[15] P. Bonami and A. Tramontani. Advances in CPLEX for mixed integer nonlinear optimization. In *International Symposium on Mathematical Programming. Pittsburgh. PA, USA*, 2015.

[16] R. Boorstyn and F. Howard. Large-Scale Network Topological Optimization. *IEEE Transactions on Communications*, 25(1):29 – 47, 1977.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization (Chapter 4)*. Cambridge University Press, 2004.

[18] Cockerham C.C. Group inbreeding and coancestry. *Genetics*, 56(1):89–104, 1967.

[19] 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.

[20] E. Cohen and J. C. Beck. Problem Difficulty and the Phase Transition in Heuristic Search. *AAAI Conference on Artificial Intelligence*, pages 780–786, 2017.

[21] B. De Backer, V. Furnon, P. Shaw, P. Kilby, and P. Prosser. Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics. *Journal of Heuristics*, 6(4):501–523, 2000.

[22] E. B. Fisher, R. P. O'Neill, and M. C. Ferris. Optimal Transmission Switching. *IEEE Transactions on Power Systems*, 23(3):1346–1355, 2008.

[23] F. Focacci, A. Lodi, and M. Milano. Cost-Based Domain Filtering. In *International Conference on Principles and Practice of Constraint Programming*, pages 189–203. Springer, 1999.

[24] F. Focacci, A. Lodi, and M. Milano. Embedding Relaxations in Global Constraints for Solving TSP and TSPTW. *Annals of Mathematics and Artificial Intelligence*, 34(4):291–311, 2002.

[25] F. Focacci, A. Lodi, and M. Milano. A Hybrid Exact Algorithm for the TSPTW. *INFORMS Journal on Computing*, 14(4):403–417, 2002.

[26] R. Fourer and D. M. Gay. Extending an algebraic modeling language to support constraint programming. *INFORMS Journal on Computing*, 14(4):322–344, 2002.

[27] J. Hallander and P. Waldmann. Optimization of selection contribution and mate allocations in monoecious tree breeding populations. *BMC Genetics*, 10(1):70, 2009.

[28] J. Hallander and P. Waldmann. Optimum contribution selection in large general tree breeding populations with an application to Scots Pine. *Theoretical and Applied Genetics*, 118(6):1133–1142, 2009.

[29] J. N. Hooker and W. J. van Hoeve. Constraint programming and operations research. *Constraints*, 23(2):172–195, 2018.

[30] P. K. Ingvarsson and H. Dahlberg. The effects of clonal forestry on genetic diversity in wild and domesticated stands of forest trees. *Scandinavian Journal of Forest Research*, pages 1–10, 2018.

[31] B. Kirkpatrick. Fast Computation of the Kinship Coefficients. *arXiv preprint arXiv:1602.04368*, 2016.

[32] 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.

[33] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

[34] J. Lee and S. Leyffer. *Mixed Integer Nonlinear Programming*, volume 154. Springer Science & Business Media, 2011.

[35] 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.

[36] 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.

[37] D. Lindgren, W. S. Libby, and F. L. Bondesson. Deployment to plantations of numbers and proportions of clones with special emphasis on maximizing gain at a constant diversity. *Theoretical and Applied Genetics*, 77(6):825–831, 1989.

[38] C. Liu, D. M. Aleman, and J. C. Beck. Modelling and Solving the Senior Transportation Problem. In *Proceedings of the Fifteenth International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR2018)*, pages 412–428. Springer, 2018.

[39] A. Martin, M. Möller, and S. Moritz. Mixed Integer Models for the Stationary Case of Gas Network Optimization. *Mathematical Programming*, 105(2–3):563–582, 2006.

[40] T.A. McRae, G.W. Dutkowski, D.J. Pilbeam, M.B. Powell, and B. Tier. Genetic evaluation using the TREEPLAN system. *Forest genetics and tree breeding in the age of genomics: Progress and future. IUFRO Joint Conference of Division*, pages 388–399, 2004.

[41] T. Meuwissen. Maximizing the response of selection with a predefined rate of inbreeding. *Journal on Animal Science*, 75(4):934–940, 1997.

[42] T. Meuwissen. GENCONT: an operational tool for controlling inbreeding in selection and conservation schemes. *Proceedings of 7th World Congress on Genetics Applied to Livestock Production*, pages 19–23, 2002.

[43] T.J. Mullin. OPSEL 2.0: A computer program for optimal selection in tree breeding. *Arbetsrapport från Skogforsk*, (954–2017), 2017.

[44] T.J. Mullin and P. Belotti. Using branch-and-bound algorithms to optimize selection of a fixed-size breeding. *Tree Genetics & Genomes*, 12(1):4, 2016.

[45] T.J. Mullin and T. Persson. Assembling optimum breeding populations for the Swedish Scots pine breeding program. *Arbetsrapport från Skogforsk*, (951-2017), 2017.

[46] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[47] A. M. Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and Computational Number Theory*, 42:75–88, 1990.

[48] G. Ottosson, E. S. Thorsteinsson, and J. N. Hooker. Mixed Global Constraints and Inference in hybrid CLP-IP solvers. *Annals of Mathematics and Artificial Intelligence*, 34(4):271–290, 2002.

[49] R. Pong-Wong and J. A. Woolliams. Optimisation of contribution of candidate parents to maximise genetic gain and restricting inbreeding using semidefinite programming. *Genetics Selection Evolution*, 39(1):3–25, 2007.

[50] A. Pruessner, M. Bussieck, S. Dirkse, and A. Meeraus. Conic programming in GAMS. In *INFORMS Annual Meeting*, pages 19–22, 2003.

[51] C.-G. Quimper. *Efficient Propagators for Global Constraints*. PhD thesis, University of Waterloo, Ontario, Canada, 2006.

[52] J.-C. Régin. A Filtering Algorithm for Constraints of Difference in CSPs. *Proceedings of the 12th National Conference on Artificial Intelligence*, 94:362–367, 1994.

[53] J.-C. Régin. Global Constraints: A Survey. In *Hybrid Optimization*, pages 63–134. Springer, 2011.

[54] O. Rosvall, C. Almqvist, D. Lindgren, and T. J. Mullin. *Breeding strategies*. Review of the Swedish tree breeding programme, Skogforsk, Uppsala, 2011.

[55] S. Sager, H. G. Bock, and G. Reinelt. Direct methods with maximal lower bound for mixed-integer optimal control problems. *Mathematical Programming*, 118(1):109–149, 2009.

[56] C. Schulte and P. J. Stuckey. Efficient Constraint Propagation Engines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(1):1–2, 12 2008.

[57] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1–4):625–653, 1999.

[58] G. Tack, C. Schulte, and G. Smolka. Generating Propagators for Finite Set Constraints. *International Conference on Principles and Practice of Constraint Programming*, pages 575–589, 2006.

[59] P. J. Teunissen and A. Kleusberg. *GPS for Geodesy*, volume 2. Springer Berlin, 1998.

[60] E. A. Thompson. Identity by descent: variation in meiosis, across genomes, and in populations. *Genetics*, 194(2):301–326, 2013.

[61] K.C. Toh, M.J. Todd, and R.H Tütüncü. SDPT3–a MATLAB software package for semidefinite programming. *Optimization Methods Software*, 11,12(1–4):545–581, 1999.

[62] T. T. Tran, T. S. Vaquero, G. Nejat, and J. C. Beck. Robots in Retirement Homes: Applying Off-the-Shelf Planning and Scheduling to a Team of Assistive Robots. *Journal of Artificial Intelligence Research*, 58:523–590, 2017.

[63] E. Tsang. Constraint Based Scheduling: Applying Constraint Programming to Scheduling Problems. *Journal of Scheduling*, 6(4):413–414, 2003.

[64] S.D.O. Turner, D. A. Romero, P.Y. Zhang, C. H. Amon, and T.C.Y. Chan. A new mathematical programming approach to optimize wind farm layouts. *Renewable Energy*, 63:674–680, 2014.

[65] P. van Emde Boas. *Another NP-complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice*. Universiteit van Amsterdam. Mathematisch Instituut, 1981.

[66] W.-J. van Hoeve and I. Katriel. Global constraints. In *Foundations of Artificial Intelligence*, volume 2, pages 169–208. Elsevier, 2006.

[67] W.-J. van Hoeve, G. Pesant, L.-M. Rousseau, and A. Sabharwal. Revisiting the Sequence Constraint. In *International Conference on Principles and Practice of Constraint Programming*, pages 620–634. Springer, 2006.

[68] M. W. Carter. The Indefinite Zero-One Quadratic Problems. *Discrete Applied Mathematics*, 7(1):23–44, 1984.

[69] C. G. Williams and O. Savolainen. Inbreeding Depression in Conifers: Implications for Breeding Strategy. *Forest Science*, 42(1):102–117, 1996.

[70] S. Wright. Coefficients of inbreeding and relationship. *The American Naturalist*, 56(645):330–338, 1922.

[71] M. Yamashita, T.J. Mullin, and S. Safarina. An efficient second-order cone programming approach for optimal selection in tree breeding. *Optimization Letters*, pages 1–15, 2018.